EL792821560

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

Application Program Interface for Network Software Platform

Inventor(s):
Brad Abrams

ATTORNEY'S DOCKET NO. MS1-863US

TECHNICAL FIELD

This invention relates to network software, such as Web applications, and to computer software development of such network software. More particularly, this invention relates to an application program interface (API) that facilitates use of a network software platform by application programs and computer hardware.

BACKGROUND

Very early on, computer software came to be categorized as "operating system" software or "application" software. Broadly speaking, an application is software meant to perform a specific task for the computer user such as solving a mathematical equation or supporting word processing. The operating system is the software that manages and controls the computer hardware. The goal of the operating system is to make the computer resources available to the application programmer while at the same time, hiding the complexity necessary to actually control the hardware.

The operating system makes the resources available via functions that are collectively known as the Application Program Interface or API. The term API is also used in reference to a single one of these functions. The functions are often grouped in terms of what resource or service they provide to the application programmer. Application software requests resources by calling individual API functions. API functions also serve as the means by which messages and information provided by the operating system are relayed back to the application software.

In addition to changes in hardware, another factor driving the evolution of operating system software has been the desire to simplify and speed application software development. Application software development can be a daunting task, sometimes requiring years of developer time to create a sophisticated program with millions of lines of code. For a popular operating system such as Microsoft Windows®, application software developers write thousands of different applications each year that utilize the operating system. A coherent and usable operating system base is required to support so many diverse application developers.

Often, development of application software can be made simpler by making the operating system more complex. That is, if a function may be useful to several different application programs, it may be better to write it once for inclusion in the operating system, than requiring dozens of software developers to write it dozens of times for inclusion in dozens of different applications. In this manner, if the operating system supports a wide range of common functionality required by a number of applications, significant savings in applications software development costs and time can be achieved.

Regardless of where the line between operating system and application software is drawn, it is clear that for a useful operating system, the API between the operating system and the computer hardware and application software is as important as efficient internal operation of the operating system itself.

Over the past few years, the universal adoption of the Internet, and networking technology in general, has changed the landscape for computer software developers. Traditionally, software developers focused on single-site software applications for standalone desktop computers, or LAN-based computers that were connected to a limited number of other computers via a local area network (LAN). Such software applications were typically referred to as "shrink wrapped" products because the software was marketed and sold in a shrink-wrapped package. The applications utilized well-defined APIs to access the underlying operating system of the computer.

As the Internet evolved and gained widespread acceptance, the industry began to recognize the power of hosting applications at various sites on the World Wide Web (or simply the "Web"). In the networked world, clients from anywhere could submit requests to server-based applications hosted at diverse locations and receive responses back in fractions of a second. These Web applications, however, were typically developed using the same operating system platform that was originally developed for standalone computing machines or locally networked computers. Unfortunately, in some instances, these applications do not adequately transfer to the distributed computing regime. The underlying platform was simply not constructed with the idea of supporting limitless numbers of interconnected computers.

To accommodate the shift to the distributed computing environment being ushered in by the Internet, Microsoft Corporation is developing a network software platform known as the ".NET" platform (read as "Dot Net"). The platform allows developers to create Web services that will execute over the Internet. Such a dynamic shift requires a new ground-up design of an entirely new API.

In response to this challenge, the inventors developed a unique set of API functions for Microsoft's .NET™ platform.

SUMMARY

An application program interface (API) provides a set of functions for application developers who build Web applications on a network platform, such as Microsoft Corporation's .NET™ platform.

BRIEF DESCRIPTION OF THE DRAWINGS

The same numbers are used throughout the drawings to reference like features.

Fig. 1 illustrates a network architecture in which clients access Web services over the Internet using conventional protocols.

Fig. 2 is a block diagram of a software architecture for Microsoft's .NET™ platform, which includes an application program interface (API).

Fig. 3 is a block diagram of unique namespaces supported by the API, as well as function classes of the various API functions.

Fig. 4 is a block diagram of an exemplary computer that may execute all or part of the software architecture.

BRIEF DESCRIPTION OF ACCOMPANYING COMPACT DISC

Accompanying this specification is a compact disc that stores a compiled HTML help file identifying the API (application program interface) for Microsoft's .NET™ network platform. The file is named "cpref.chm" and was created on June 8, 2001. It is 30.81 Mbytes in size. The file can be executed on a Windows®-based computing device (e.g., IBM-PC, or equivalent) that executes a Windows®-brand operating system (e.g., Windows® NT, Windows® 98, Windows® 2000, etc.). The compiled HTML help file stored on the compact disk is hereby incorporated by reference.

Additionally, the APIs contained in the compiled HTML help file are also provided in approximately 100 separate text files named "NamespaceName.txt". The text files comply with the ASCII format.

The compact disc itself is a CD-ROM, and conforms to the ISO 9660 standard.

DETAILED DESCRIPTION

This disclosure addresses an application program interface (API) for a network platform upon which developers can build Web applications and services. More particularly, an exemplary API is described for the .NET™ platform created by Microsoft Corporation. The .NET™ platform is a software platform for Web services and Web applications implemented in the distributed computing environment. It represents the next generation of Internet computing, using open communication standards to communicate among loosely coupled Web services that are collaborating to perform a particular task.

In the described implementation, the .NET™ platform utilizes XML (extensible markup language), an open standard for describing data. XML is managed by the World Wide Web Consortium (W3C). XML is used for defining data elements on a Web page and business-to-business documents. XML uses a similar tag structure as HTML; however, whereas HTML defines how elements are displayed, XML defines what those elements contain. HTML uses predefined tags, but XML allows tags to be defined by the developer of the page. Thus, virtually any data items can be identified, allowing Web pages to function like database records. Through the use of XML and other open protocols, such as Simple Object Access Protocol (SOAP), the .NET™ platform allows integration of a wide range of services that can be tailored to the needs of the user. Although the embodiments described herein are described in conjunction with XML and other open standards, such are not required for the operation of the claimed invention. Other equally viable technologies will suffice to implement the inventions described herein.

As used herein, the phrase application program interface or API includes traditional interfaces that employ method or function calls, as well as remote calls (e.g., a proxy, stub relationship) and SOAP/XML invocations.

EXEMPLARY NETWORK ENVIRONMENT

Fig. 1 shows a network environment 100 in which a network platform, such as the .NET™ platform, may be implemented. The network environment 100 includes representative Web services 102(1), ..., 102(N), which provide services that can be accessed over a network 104 (e.g., Internet). The Web services, referenced generally as number 102, are programmable application components that are reusable and interact programmatically over the network 104, typically through industry standard Web protocols, such as XML, SOAP, WAP (wireless application protocol), HTTP (hypertext transport protocol), and SMTP (simple mail transfer protocol) although other means of interacting with the Web services over the network may also be used, such as Remote Procedure Call (RPC) or object broker type technology. A Web service can be self-describing and is often defined in terms of formats and ordering of messages.

Web services 102 are accessible directly by other services (as represented by communication link 106) or a software application, such as Web application 110 (as represented by communication links 112 and 114). Each Web service 102 is illustrated as including one or more servers that execute software to handle requests for particular services. Such services often maintain databases that store information to be served back to requesters. Web services may be configured to perform any one of a variety of different services. Examples of Web services include login verification, notification, database storage, stock quoting, location directories, mapping, music, electronic wallet, calendar/scheduler, telephone listings, news and information, games, ticketing, and so on. The Web services can be combined with each other and with other applications to build intelligent interactive experiences.

The network environment 100 also includes representative client devices 120(1), 120(2), 120(3), 120(4), ..., 120(M) that utilize the Web services 102 (as represented by communication link 122) and/or the Web application 110 (as represented by communication links 124, 126, and 128). The clients may communicate with one another using standard protocols as well, as represented by an exemplary XML link 130 between clients 120(3) and 120(4).

The client devices, referenced generally as number 120, can be implemented many different ways. Examples of possible client implementations include, without limitation, portable computers, stationary computers, tablet PCs, televisions/set-top boxes, wireless communication devices, personal digital assistants, gaming consoles, printers, photocopiers, and other smart devices.

The Web application 110 is an application designed to run on the network platform and may utilize the Web services 102 when handling and servicing requests from clients 120. The Web application 110 is composed of one or more software applications 130 that run atop a programming framework 132, which are executing on one or more servers 134 or other computer systems. Note that a portion of Web application 110 may actually reside on one or more of clients 120. Alternatively, Web application 110 may coordinate with other software on clients 120 to actually accomplish its tasks.

The programming framework 132 is the structure that supports the applications and services developed by application developers. It permits multi-language development and seamless integration by supporting multiple languages. It supports open protocols, such as SOAP, and encapsulates the underlying operating system and object model services. The framework provides a robust and secure execution environment for the multiple programming languages and offers secure, integrated class libraries.

The framework 132 is a multi-tiered architecture that includes an application program interface (API) layer 142, a common language runtime (CLR) layer 144, and an operating system/services layer 146. This layered architecture allows updates and modifications to various layers without impacting other portions of the framework. A common language specification (CLS) 140 allows designers of various languages to write code that is able to access underlying library functionality. The specification 140 functions as a contract between language designers and library designers that can be used to promote language interoperability. By adhering to the CLS, libraries written in one language can be directly accessible to code modules written in other languages to achieve seamless integration between code modules written in one language and code modules written in another language. One exemplary detailed implementation of a CLS is described in an ECMA standard created by participants in ECMA TC39/TG3. The reader is directed to the ECMA web site at www.ecma.ch.

The API layer 142 presents groups of functions that the applications 130 can call to access the resources and services provided by layer 146. By exposing the API functions for a network platform, application developers can create Web applications for distributed computing systems that make full use of the network resources and other Web services, without needing to understand the complex interworkings of how those network resources actually operate or are made available. Moreover, the Web applications can be written in any number of programming languages, and translated into an intermediate language supported by the common language runtime 144 and included as part of the common language specification 140. . In this way, the API layer 142 can provide methods for a wide and diverse variety of applications.

Additionally, the framework 132 can be configured to support API calls placed by remote applications executing remotely from the servers 134 that host the framework. Representative applications 148(1) and 148(2) residing on clients 120(3) and 120(M), respectively, can use the API functions by making calls directly, or indirectly, to the API layer 142 over the network 104.

The framework may also be implemented at the clients. Client 120(3) represents the situation where a framework 150 is implemented at the client. This framework may be identical to server-based framework 132, or modified for client purposes. Alternatively, the client-based framework may be condensed in the event that the client is a limited or dedicated function device, such as a cellular phone, personal digital assistant, handheld computer, or other communication/computing device.

DEVELOPERS' PROGRAMMING FRAMEWORK

Fig. 2 shows the programming framework 132 in more detail. The common language specification (CLS) layer 140 supports applications written in a variety of languages 130(1), 130(2), 130(3), 130(4), ..., 130(K). Such application languages include Visual Basic, C++, C#, COBOL, Jscript, Perl, Eiffel, Python, and so on. The common language specification 140 specifies a subset of features or rules about features that, if followed, allow the various languages to communicate. For example, some languages do not support a given type (e.g., an "int*" type) that might otherwise be supported by the common language runtime 144. In this case, the common language specification 140 does not include the type. On the other hand, types that are supported by all or most languages (e.g., the "int[]" type) is included in common language specification 140 so library developers are free to use it and are assured that the languages can handle it. This ability to communicate results in seamless integration between code modules written in one language and code modules written in another language. Since different languages are particularly well suited to particular tasks, the seamless integration between languages allows a developer to select a particular language for a particular code module with the ability to use that code module with modules written in different languages. The common language runtime 144 allow seamless multi-language development, with cross language inheritance, and provide a robust and secure execution environment for the multiple programming languages. For more information on the common language specification 140 and the common language runtime 144, the reader is directed to co-pending applications entitled "Method and System for Compiling Multiple Languages", filed 6/21/2000 (serial number 09/598,105) and "Unified Data Type System and Method" filed 7/10/2000 (serial number 09/613,289), which are incorporated by reference.

The framework 132 encapsulates the operating system 146(1) (e.g., Windows®-brand operating systems) and object model services 146(2) (e.g., Component Object Model (COM) or Distributed COM). The operating system 146(1) provides conventional functions, such as file management, notification, event handling, user interfaces (e.g., windowing, menus, dialogs, etc.), security, authentication, verification, processes and threads, memory management, and so on. The object model services 146(2) provide interfacing with other objects to perform various tasks. Calls made to the API layer 142 are handed to the common language runtime layer 144 for local execution by the operating system 146(1) and/or object model services 146(2).

The API 142 groups API functions into multiple namespaces. Namespaces essentially define a collection of classes, interfaces, delegates, enumerations, and structures, which are collectively called "types", that provide a specific set of related functionality. A class represents managed heap allocated data that has reference assignment semantics. A delegate is an object oriented function pointer. An enumeration is a special kind of value type that represents named constants. A structure represents static allocated data that has value assignment semantics. An interface defines a contract that other types can implement.

By using namespaces, a designer can organize a set of types into a hierarchical namespace. The designer is able to create multiple groups from the set of types, with each group containing at least one type that exposes logically related functionality. In the exemplary implementation, the API 142 is organized into four root namespaces: a first namespace 200 for Web applications, a second namespace 202 for client applications, a third namespace 204 for data and XML, and a fourth namespace 206 for base class libraries (BCLs). Each group can then be assigned a name. For instance, types in the Web applications namespace 200 are assigned the name "Web", and types in the data and XML namespace 204 can be assigned names "Data" and "XML" respectively. The named groups can be organized under a single "global root" namespace for system level APIs, such as an overall System namespace. By selecting and prefixing a top level identifier, the types in each group can be easily referenced by a hierarchical name that includes the selected top level identifier prefixed to the name of the group containing the type. For instance, types in the Web applications namespace 200 can be referenced using the hierarchical name "System.Web". In this way, the individual namespaces 200, 202, 204, and 206 become major branches off of the System namespace and can carry a designation where the individual namespaces are prefixed with a designator, such as a "System." prefix.

The Web applications namespace 200 pertains to Web based functionality, such as dynamically generated Web pages (e.g., Microsoft's Active Server Pages (ASP)). It supplies types that enable browser/server communication. The client applications namespace 202 pertains to drawing and client side UI functionality. It supplies types that enable drawing of two-dimensional (2D), imaging, and printing, as well as the ability to construct window forms, menus, boxes, and so on.

The data and XML namespace 204 relates to connectivity to data sources and XML functionality. It supplies classes, interfaces, delegates, and enumerations that enable security, specify data types, and serialize objects into XML format documents or streams. The base class libraries (BCL) namespace

206 pertains to basic system and runtime functionality. It contains the fundamental types and base classes that define commonly-used value and reference data types, events and event handlers, interfaces, attributes, and processing exceptions.

In addition to the framework 132, programming tools 210 are provided to assist the developer in building Web services and/or applications. One example of the programming tools 200 is Visual Studio™, a multi-language suite of programming tools offered by Microsoft Corporation.

ROOT API NAMESPACES

Fig. 3 shows the API 142 and its four root namespaces in more detail. In one embodiment, the namespaces are identified according to a hierarchical naming convention in which strings of names are concatenated with periods. For instance, the Web applications namespace 200 is identified by the root name "System.Web". Within the "Sytem.Web" namespace is another namespace for Web services, identified as "System.Web.Services", which further identifies another namespace for a description known as "System.Web.Services.Description". With this naming convention in mind, the following provides a general overview of selected namespaces of the API 142, although other naming conventions could be used with equal effect.

The Web applications namespace 200 ("System.Web") defines additional namespaces, including:

- A services namespace 300 ("System.Web.Services") containing classes that enable a developer to build and use Web services. The services namespace 300 defines additional namespaces, including a description namespace 302 ("System.Web.Services.Description") containing classes that enable a developer to publicly describe a Web service via a service description language (such as WSDL, a specification available from the W3C), a discovery namespace 304 ("System.Web.Services.Discovery") containing classes that allow Web service consumers to locate available Web Services on a Web server, and a protocols namespace 306 ("System.Web.Services.Protocols") containing classes that define the protocols used to transmit data across a network during communication between Web service clients and the Web service itself.

- A caching namespace 308 ("System.Web.Caching") containing classes that enable developers to decrease Web application response time through temporarily caching frequently used resources on the server. This includes ASP.NET pages, web services, and user controls. (ASP.NET is the updated version of Microsoft's ASP technology.) Additionally, a cache dictionary is available for developers to store frequently used resources, such as hash tables and other data structures.
- A configuration namespace 310 ("System.Web.Configuration") containing classes that are used to read configuration data in for an application.
- A UI namespace 312 ("System.Web.UI") containing types that allow developers to create controls and pages that will appear in Web

applications as user interfaces on a Web page. This namespace includes the control class, which provides all web based controls, whether those encapsulating HTML elements, higher level Web controls, or even custom User controls, with a common set of functionality. Also provided are classes which provide the web forms server controls data binding functionality, the ability to save the view state of a given control or page, as well as parsing functionality for both programmable and literal controls. Within the UI namespace 312 are two additional namespaces: an HTML controls namespace 314 ("System.Web.UI.HtmlControls") containing classes that permit developers to interact with types that encapsulates html 3.2 elemtents create HTML controls, and a Web controls namespace 316 ("System.Web.UI.WeblControls") containing classes that allow developers to create higher level Web controls.

- A security namespace 318 ("System.Web.Security") containing classes used to implement security in web server applications, such as basic authentication, challenge response authentication, and role based authentication.
- A session state namespace 320 ("System.Web.SessionState") containing classes used to access session state values (i.e., data that lives across requests for the lifetime of the session) as well as session-level settings and lifetime management methods.

The client applications namespace 202 is composed of two namespaces:

- A windows forms namespace 322 ("System.Windows.Forms") containing classes for creating Windows®-based client applications that take full advantage of the rich user interface features available in the Microsoft Windows® operating system, such as the ability to drag and drop screen elements. Such classes may include wrapped APIs available in the Microsoft Windows® operating system that are used in a windowing UI environment. Within this namespace are a design namespace 324 ("System.Windows.Forms.Design") that contains classes to extend design-time support for Windows forms and a component model namespace 326 ("System.Windows.Forms.ComponentModel") that contains the windows form implementation of the general component model defined in System.ComponentModel. This namespace contains designer tools, such as Visual Studio, which offer a rich experience for developers at design time.
- A drawing namespace 328 ("System.Drawing") containing classes for graphics functionality. The drawing namespace 328 includes a 2D drawing namespace 330 ("System.Drawing.Drawing2D") that contains classes and enumerations to provide advanced 2-dimmensional and vector graphics functionality, an imaging namespace 332 ("System.Drawing.Imaging") that contains classes for advanced imaging functionality, a printing namespace 334 ("System.Drawing.Printing") that contains classes to permit developers to customize printing, and a text namespace 336

("System.Drawing.Text") that contains classes for advanced typography functionality.

The data and XML namespace 204 is composed of two namespaces:

- A data namespace 340 ("System.Data") containing classes that enable developers to build components that efficiently manage data from multiple data sources. It implements an architecture that, in a disconnected scenario (such as the Internet), provides tools to request, update, and reconcile data in multiple tier systems. The data namespace 340 includes a common namespace 342 that contains types shared by data providers. A data provider describes a collection of types used to access a data source, such as a database, in the managed space. The data namespace 340 also includes an OLE DB namespace 344 that contains types pertaining to data used in object-oriented databases (e.g., Microsoft's SQL Server), and a SQL client namespace 346 that contains types pertaining to data used by SQL clients. The data namespace also includes a SQL types namespace 348 ("System.Data.SqlTypes") that contains classes for native data types within Microsoft's SQL Server. The classes provide a safer, faster alternative to other data types. Using the objects within this namespace helps prevent type conversion errors caused in situations where loss of precision could occur. Because other data types are converted to and from SQL types behind the scenes, explicitly creating and using objects within this namespace results in faster code as well.

- An XML namespace 350 ("System.XML") containing classes that provide standards-based support for processing XML. The supported standards include XML (e.g., version 1.0), XML Namespaces (both stream level and DOM), XML Schemas, XPath expressions, XSL/T transformations, DOM Level 2 Core, and SOAP (e.g., version 1.1). The XML namespace 350 includes an XSLT namespace 352 ("System.XML.Xsl") that contains classes and enumerations to support XSLT (Extensible Stylesheet Language Transformations), an Xpath namespace 354 ("System.XML.Xpath") that contains an XPath parser and evaluation engine, and a serialization namespace 356 ("System.XML.Serialization") that contains classes used to serialize objects into XML format documents or streams.

The base class library namespace 206 ("System") includes the following namespaces:

- A collections namespace 360 ("System.Collections") containing interfaces and classes that define various collections of objects, such as lists, queues, arrays, hash tables and dictionaries.
- A configuration namespace 362 ("System.Configuration") containing classes and interfaces that allow developers to programmatically access configuration settings and handle errors in configuration files.

- A diagnostics namespace 364 ("System.Diagnostics") containing classes that are used to debug applications and to trace code execution. The namespace allows developers to start system processes, read and write to event logs, and monitor system performance using performance counters.
- A globalization namespace 366 ("System.Globalization") containing classes that define culture-related information, including the language, the country/region, the calendars in use, the format patterns for dates, currency and numbers, and the sort order for strings.
- An I/O namespace 368 ("System.IO") containing the infrastructure pieces to operate with the intput/output of data streams, files, and directories. This namespace includes a model for working with streams of bytes, higher level readers and writers which consume those bytes, various constructions or implementations of the streams (e.g., FileStream and MemoryStream) and, a set of utility classes for working with files and directories.
- A net namespace 370 ("System.Net") providing an extensive set of classes for building network-enabled application, referred to as the Net Class Libraries (NCL). One element to the design of the Net Class Libraries is an extensible, layered approach to exposing networking functionality. The NCL stack contains three basic layers. A base layer (System.Net.Socket) provides access to an interface to TCP/IP, the communications protocol of UNIX networks and the Internet. One example of such an interface is the "WinSock

API" from Microsoft Corporation. The next layer is the Transport Protocol classes, which support such transport protocols as TCP and UDP. Developers may write their own protocol classes to provide support for protocols such as IGMP and ICMP. The third layer is the Web request, which provides an abstract factory pattern for the creation of other protocol classes. The NCL provides implementations for Hyper Text Transport Protocol (HTTP).

- A reflection namespace ("System.Reflection") 372 containing types that provide a managed view of loaded types, methods, and fields, with the ability to dynamically create and invoke types.
- A resources namespace 374 ("System.Resources") containing classes and interfaces that allow developers to create, store and manage various culture-specific resources used in an application.
- A security namespace 376 ("System.Security") supporting the underlying structure of the security system, including interfaces, attributes, exceptions, and base classes for permissions.
- A service process namespace 378 ("System.ServiceProcess") containing classes that allow developers to install and run services. Services are long-running executables that run without a user interface. They can be installed to run under a system account that enables them to be started at computer reboot. Services whose implementation is derived from processing in one class can define specific behavior for start, stop, pause, and continue commands, as well as behavior to take when the system shuts down.

- A text namespace 380 ("System.Text") containing classes representing various types of encodings (e.g., ASCII, Unicode, UTF-7, and UTF-8), abstract base classes for converting blocks of characters to and from blocks of bytes, and a helper class that manipulates and formats string objects without creating intermediate instances.
- A threading namespace 382 ("System.Threading") containing classes and interfaces that enable multi-threaded programming. The threading namespace includes a ThreadPool class that manages groups of threads, a Timer class that enables a delegate to be called after a specified amount of time, and a Mutex class for synchronizing mutually-exclusive threads. This namespace also provides classes for thread scheduling, wait notification, and deadlock resolution.
- A runtime namespace 384 ("System.Runtime") containing multiple namespaces concerning runtime features, including an interoperation services namespace 386 ("System.Runtime.InteropServices") that contains a collection of classes useful for accessing COM objects. The types in the InteropServices namespace fall into the following areas of functionality: attributes, exceptions, managed definitions of COM types, wrappers, type converters, and the Marshal class. The runtime namespace 384 further includes a remoting namespace 388 ("System.Runtime.Remoting") that contains classes and interfaces allowing developers to create and configure distributed applications. Another namespace within the runtime namespace 384 is a

serialization namespace 390 ("System.Runtime.Serialization") that contains classes used for serializing and deserializing objects. Serialization is the process of converting an object or a graph of objects into a linear sequence of bytes for either storage or transmission to another location.

The web applications namespace 200 ("System.Web") defines several additional namespaces, including the services namespace 300 ("System.Web.Services"), a caching namespace 308 ("System.Web.Caching"), a configuration namespace 310 ("System.Web.Configuration"), a UI namespace 312 ("System.Web.UI"), a security namespace 318 ("System.Web.Security"), and a session state namespace 320 ("System.Web.SessionState"). In general, the web applications namespace 200 supplies tools that enable browser-server communication.

The services namespace 300 contains classes that allow developers to build and use various web services. The services namespace includes a web service class that defines a base class for web services and a web method attribute class that allows a method to be programmatically exposed over the web.

The UI namespace 312 contains classes that allow a user to create HTML server controls on a web page. These HTML server controls execute on the server and map to standard HTML tags. The UI namespace also contains classes that allow a user to create web server controls on a web page. These web server controls run on the web server and include form controls, such as buttons and text boxes.

The web applications namespace 200 also includes classes for manipulating cookies, transferring files, handling exception information, and controlling an output cache. Specific details regarding the System.Web namespace are provided below.

System.Web

Description

The System.Web namespace supplies classes and interfaces that enable browser/server communication. This namespace includes the HTTPRequest class that provides extensive information about the current HTTP request, the HTTPResponse class that manages HTTP output to the client, and the HTTPServerUtility object that provides access to server-side utilities and processes. System.Web also includes classes for cookie manipulation, file transfer, exception information, and output cache control.

BeginEventHandler delegate (System.Web)

Description

EndEventHandler delegate (System.Web)

Description

HttpWorkerRequest.EndOfSendNotification delegate (System.Web)

Description

HttpApplication class (System.Web)

Description

Defines the methods, properties, and events common to all application objects within an ASP.NET application.

Constructors:

HttpApplication

Example Syntax:

[C#] public HttpApplication();

[C++] public: HttpApplication();

[VB] Public Sub New()

[JScript] public function HttpApplication();

Properties:

Application

[C#] public HttpApplicationState Application {get;}

[C++] public: __property HttpApplicationState* get_Application();

[VB] Public ReadOnly Property Application As HttpApplicationState

[JScript] public function get Application() : HttpApplicationState;

Description

Gets a reference to an **HTTPApplication** state bag instance.

Context

[C#] public HttpContext Context {get;}

[C++] public: __property HttpContext* get_Context();

[VB] Public ReadOnly Property Context As HttpContext

[JScript] public function get Context() : HttpContext;

Description

Gets the **HTTPRuntime** -provided context object that provides access to additional pipeline-module exposed objects.

Events

[C#] protected EventHandlerList Events {get;}

[C++] protected: __property EventHandlerList* get_Events();

[VB] Protected ReadOnly Property Events As EventHandlerList

[JScript] protected function get Events() : EventHandlerList;

Description

Modules

[C#] public HttpModuleCollection Modules {get;}

[C++] public: __property HttpModuleCollection* get_Modules();
[VB] Public ReadOnly Property Modules As HttpModuleCollection
[JScript] public function get Modules() : HttpModuleCollection;

Description

Gets the collection of **HTTPModules** configured for the current application.

Request

[C#] public HttpRequest Request {get;}
[C++] public: __property HttpRequest* get_Request();
[VB] Public ReadOnly Property Request As HttpRequest
[JScript] public function get Request() : HttpRequest;

Description

Gets the intrinsic object that provides access to incoming **HttpRequest** data.

Response

[C#] public HttpResponse Response {get;}
[C++] public: __property HttpResponse* get_Response();
[VB] Public ReadOnly Property Response As HttpResponse
[JScript] public function get Response() : HttpResponse;

Description

The intrinsic object that allows transmission of **HttpResponse** data to a client.

Server

```
[C#] public HttpServerUtility Server {get;}
[C++] public: __property HttpServerUtility* get_Server();
[VB] Public ReadOnly Property Server As HttpServerUtility
[JScript] public function get Server() : HttpServerUtility;
```

Description

Gets the intrinsic **Server** object.

Session

```
[C#] public HttpSessionState Session {get;}
[C++] public: __property HttpSessionState* get_Session();
[VB] Public ReadOnly Property Session As HttpSessionState
[JScript] public function get Session() : HttpSessionState;
```

Description

Gets the intrinsic **Session** object that provides access to session data.

Site

```
[C#] public ISite Site {get; set;}
[C++] public: __property ISite* get_Site();public: __property void set_Site(ISite*);
```

[VB] Public Property Site As ISite
[JScript] public function get Site() : ISite;public function set Site(ISite);

Description

User

[C#] public IPrincipal User {get;}
[C++] public: __property IPrincipal* get_User();
[VB] Public ReadOnly Property User As IPrincipal
[JScript] public function get User() : IPrincipal;

Description

Gets the **User** intrinsic object.

[C#] public event EventHandler AcquireRequestState;
[C++] public: __event EventHandler* AcquireRequestState;
[VB] Public Event AcquireRequestState As EventHandler

Description

[C#] public event EventHandler AuthenticateRequest;
[C++] public: __event EventHandler* AuthenticateRequest;
[VB] Public Event AuthenticateRequest As EventHandler

Description

```
[C#]      public        event       EventHandler        AuthorizeRequest;
[C++]     public:       __event     EventHandler*       AuthorizeRequest;
[VB]      Public        Event       AuthorizeRequest    As    EventHandler
```

Description

```
[C#]      public        event       EventHandler        BeginRequest;
[C++]     public:       __event     EventHandler*       BeginRequest;
[VB]      Public        Event       BeginRequest        As    EventHandler
```

Description

```
[C#]      public           event       EventHandler        Disposed;
[C++]     public:          __sealed    __event     EventHandler*    Disposed;
[VB]      NotOverridable   Public      Event       Disposed    As    EventHandler
```

Description

[C#] public event EventHandler EndRequest;
[C++] public: __event EventHandler* EndRequest;
[VB] Public Event EndRequest As EventHandler

Description

[C#] public event EventHandler Error;
[C++] public: __event EventHandler* Error;
[VB] Public Event Error As EventHandler

Description

[C#] public event EventHandler PostRequestHandlerExecute;
[C++] public: __event EventHandler* PostRequestHandlerExecute;
[VB] Public Event PostRequestHandlerExecute As EventHandler

Description

[C#] public event EventHandler PreRequestHandlerExecute;
[C++] public: __event EventHandler* PreRequestHandlerExecute;
[VB] Public Event PreRequestHandlerExecute As EventHandler

Description

```
[C#]    public    event    EventHandler    PreSendRequestContent;
[C++]   public:   __event  EventHandler*   PreSendRequestContent;
[VB]    Public    Event    PreSendRequestContent    As    EventHandler
```

Description

```
[C#]    public    event    EventHandler    PreSendRequestHeaders;
[C++]   public:   __event  EventHandler*   PreSendRequestHeaders;
[VB]    Public    Event    PreSendRequestHeaders    As    EventHandler
```

Description

```
[C#]    public    event    EventHandler    ReleaseRequestState;
[C++]   public:   __event  EventHandler*   ReleaseRequestState;
[VB]    Public    Event    ReleaseRequestState    As    EventHandler
```

Description

[C#] public event EventHandler ResolveRequestCache;
[C++] public: __event EventHandler* ResolveRequestCache;
[VB] Public Event ResolveRequestCache As EventHandler

Description

[C#] public event EventHandler UpdateRequestCache;
[C++] public: __event EventHandler* UpdateRequestCache;
[VB] Public Event UpdateRequestCache As EventHandler

Description

Methods:

AddOnAcquireRequestStateAsync

[C#] public void AddOnAcquireRequestStateAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnAcquireRequestStateAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnAcquireRequestStateAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnAcquireRequestStateAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnAuthenticateRequestAsync

[C#] public void AddOnAuthenticateRequestAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnAuthenticateRequestAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnAuthenticateRequestAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnAuthenticateRequestAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnAuthorizeRequestAsync

[C#] public void AddOnAuthorizeRequestAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnAuthorizeRequestAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnAuthorizeRequestAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnAuthorizeRequestAsync(bh : BeginEventHandler,

eh : EndEventHandler);

Description

AddOnBeginRequestAsync

[C#] public void AddOnBeginRequestAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnBeginRequestAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnBeginRequestAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnBeginRequestAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnEndRequestAsync

[C#] public void AddOnEndRequestAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnEndRequestAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnEndRequestAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)

[JScript] public function AddOnEndRequestAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnPostRequestHandlerExecuteAsync

[C#] public void AddOnPostRequestHandlerExecuteAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnPostRequestHandlerExecuteAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnPostRequestHandlerExecuteAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnPostRequestHandlerExecuteAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnPreRequestHandlerExecuteAsync

[C#] public void AddOnPreRequestHandlerExecuteAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnPreRequestHandlerExecuteAsync(BeginEventHandler* bh, EndEventHandler* eh);

[VB] Public Sub AddOnPreRequestHandlerExecuteAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnPreRequestHandlerExecuteAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnReleaseRequestStateAsync

[C#] public void AddOnReleaseRequestStateAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnReleaseRequestStateAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnReleaseRequestStateAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnReleaseRequestStateAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnResolveRequestCacheAsync

[C#] public void AddOnResolveRequestCacheAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnResolveRequestCacheAsync(BeginEventHandler* bh,

EndEventHandler* eh);
[VB] Public Sub AddOnResolveRequestCacheAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnResolveRequestCacheAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

AddOnUpdateRequestCacheAsync

[C#] public void AddOnUpdateRequestCacheAsync(BeginEventHandler bh, EndEventHandler eh);
[C++] public: void AddOnUpdateRequestCacheAsync(BeginEventHandler* bh, EndEventHandler* eh);
[VB] Public Sub AddOnUpdateRequestCacheAsync(ByVal bh As BeginEventHandler, ByVal eh As EndEventHandler)
[JScript] public function AddOnUpdateRequestCacheAsync(bh : BeginEventHandler, eh : EndEventHandler);

Description

CompleteRequest

[C#] public void CompleteRequest();
[C++] public: void CompleteRequest();

[VB] Public Sub CompleteRequest()
[JScript] public function CompleteRequest();

Description

Dispose

[C#] public virtual void Dispose();
[C++] public: virtual void Dispose();
[VB] Overridable Public Sub Dispose()
[JScript] public function Dispose();

Description

Cleans up the instance variables of an **HttpModule.**

The **System.Web.HttpApplication.Request**, **System.Web.HttpApplication.Response**, **System.Web.HttpApplication.Session** and **System.Web.HttpApplication.Application** properties are not available for use at the time **System.Web.HttpApplication.Dispose** is executed.

GetVaryByCustomString

[C#] public virtual string GetVaryByCustomString(HttpContext context, string custom);
[C++] public: virtual String* GetVaryByCustomString(HttpContext* context, String* custom);

[VB] Overridable Public Function GetVaryByCustomString(ByVal context As HttpContext, ByVal custom As String) As String
[JScript] public function GetVaryByCustomString(context : HttpContext, custom : String) : String;

Description

Init

[C#] public virtual void Init();
[C++] public: virtual void Init();
[VB] Overridable Public Sub Init()
[JScript] public function Init();

Description

Initializes **HttpModule** instance variables and register event handlers with the hosting Application.

IHttpAsyncHandler.BeginProcessRequest

[C#] IAsyncResult IHttpAsyncHandler.BeginProcessRequest(HttpContext context, AsyncCallback cb, object extraData);
[C++] IAsyncResult* IHttpAsyncHandler::BeginProcessRequest(HttpContext* context, AsyncCallback* cb, Object* extraData);
[VB] Function BeginProcessRequest(ByVal context As HttpContext, ByVal cb As AsyncCallback, ByVal extraData As Object) As IAsyncResult Implements

IHttpAsyncHandler.BeginProcessRequest

[JScript] function IHttpAsyncHandler.BeginProcessRequest(context : HttpContext, cb : AsyncCallback, extraData : Object) : IAsyncResult;

IHttpAsyncHandler.EndProcessRequest

[C#] void IHttpAsyncHandler.EndProcessRequest(IAsyncResult result);

[C++] void IHttpAsyncHandler::EndProcessRequest(IAsyncResult* result);

[VB] Sub EndProcessRequest(ByVal result As IAsyncResult) Implements IHttpAsyncHandler.EndProcessRequest

[JScript] function IHttpAsyncHandler.EndProcessRequest(result : IAsyncResult);

IHttpHandler.ProcessRequest

[C#] void IHttpHandler.ProcessRequest(HttpContext context);

[C++] void IHttpHandler::ProcessRequest(HttpContext* context);

[VB] Sub ProcessRequest(ByVal context As HttpContext) Implements IHttpHandler.ProcessRequest

[JScript] function IHttpHandler.ProcessRequest(context : HttpContext);

HttpApplicationState class (System.Web)

ToString

Description

Enables sharing of global information across multiple sessions and requests within an ASP.NET application.

An ASP.NET application is the sum of all files, pages, handlers, modules, and code within the scope of a virtual directory and its subdirectories on a single web server.

AllKeys

ToString

[C#] public string[] AllKeys {get;}
[C++] public: __property String* get_AllKeys();
[VB] Public ReadOnly Property AllKeys As String ()
[JScript] public function get AllKeys() : String[];

Description

Gets the access keys in the **System.Web.HttpApplicationState** collection.

Contents

ToString

[C#] public HttpApplicationState Contents {get;}
[C++] public: __property HttpApplicationState* get_Contents();
[VB] Public ReadOnly Property Contents As HttpApplicationState
[JScript] public function get Contents() : HttpApplicationState;

Description

Gets a reference to the **System.Web.HttpApplicationState** object.

This property provides compatibility with earlier versions of ASP.

Count

ToString

[C#] public override int Count {get;}
[C++] public: __property virtual int get_Count();
[VB] Overrides Public ReadOnly Property Count As Integer
[JScript] public function get Count() : int;

Description

Gets the number of objects in the **System.Web.HttpApplicationState** collection.

IsReadOnly

Item

ToString

System.Web.HttpApplicationState

Description

Gets the value of a single **System.Web.HttpApplicationState** object by name. The name of the object in the collection.

Item

ToString

[C#] public object this[int index] {get;}
[C++] public: __property Object* get_Item(int index);
[VB] Public Default ReadOnly Property Item(ByVal index As Integer) As Object
[JScript] returnValue = HttpApplicationStateObject.Item(index);

Description

Gets a single **System.Web.HttpApplicationState** object by index. The numerical index of the object in the collection.

Keys

StaticObjects

ToString

Description

Gets all objects declared via an tag within the ASP.NET application.

Application objects are defined in the Global.asax file.

Add

```
[C#] public void Add(string name, object value);
[C++] public: void Add(String* name, Object* value);
[VB] Public Sub Add(ByVal name As String, ByVal value As Object)
[JScript] public function Add(name : String, value : Object);
```

Description

Adds a new object to the **System.Web.HttpApplicationState** collection. The name of the object to be added to the collection. The value of the object.

Clear

```
[C#] public void Clear();
```

[C++] public: void Clear();
[VB] Public Sub Clear()
[JScript] public function Clear();

Description

Removes all objects from an **System.Web.HttpApplicationState** collection.

Get

[C#] public object Get(int index);
[C++] public: Object* Get(int index);
[VB] Public Function Get(ByVal index As Integer) As Object
[JScript] public function Get(index : int) : Object;

Description

Gets an **System.Web.HttpApplicationState** object by numerical index. *Return Value:* The object referenced by *index* . The index of the application state object.

Get

[C#] public object Get(string name);
[C++] public: Object* Get(String* name);
[VB] Public Function Get(ByVal name As String) As Object
[JScript] public function Get(name : String) : Object; Gets an **System.Web.HttpApplicationState** object by name or index.

Description

Gets an **System.Web.HttpApplicationState** object by name. *Return Value:* The object referenced by *name* .

The following example returns an object named MyAppVar1 from the **System.Web.HttpApplicationState** collection of the intrinsic **System.Web.HttpContext.Application** object and copies it to a new object variable. The name of the object.

GetKey

```
[C#] public string GetKey(int index);
[C++] public: String* GetKey(int index);
[VB] Public Function GetKey(ByVal index As Integer) As String
[JScript] public function GetKey(index : int) : String;
```

Description

Gets an **System.Web.HttpApplicationState** object name by index. *Return Value:* The name under which the application state object was saved. The index of the application state object.

Lock

```
[C#] public void Lock();
[C++] public: void Lock();
[VB] Public Sub Lock()
[JScript] public function Lock();
```

Description

Locks access to an **System.Web.HttpApplicationState** variable to facilitate access synchronization.

Remove

```
[C#] public void Remove(string name);
[C++] public: void Remove(String* name);
[VB] Public Sub Remove(ByVal name As String)
[JScript] public function Remove(name : String);
```

Description

Removes the named object from an **System.Web.HttpApplicationState** collection. The name of the object to be removed from the collection.

RemoveAll

```
[C#] public void RemoveAll();
[C++] public: void RemoveAll();
[VB] Public Sub RemoveAll()
[JScript] public function RemoveAll();
```

Description

Removes all objects from an **System.Web.HttpApplicationState** collection.

System.Web.HttpApplicationState.RemoveAll is an internal call to **System.Web.HttpApplicationState.Clear** .

RemoveAt

[C#] public void RemoveAt(int index);
[C++] public: void RemoveAt(int index);
[VB] Public Sub RemoveAt(ByVal index As Integer)
[JScript] public function RemoveAt(index : int); Removes an object from the application state collection by name.

Set

[C#] public void Set(string name, object value);
[C++] public: void Set(String* name, Object* value);
[VB] Public Sub Set(ByVal name As String, ByVal value As Object)
[JScript] public function Set(name : String, value : Object);

Description

Updates the value of an object in an **System.Web.HttpApplicationState** collection. The name of the object to be updated. The updated value of the object.

UnLock

[C#] public void UnLock();
[C++] public: void UnLock();
[VB] Public Sub UnLock()
[JScript] public function UnLock();

Description

Unlocks access to an **System.Web.HttpApplicationState** variable to facilitate access synchronization.

HttpBrowserCapabilities class (System.Web)

UnLock

Description

Enables the server to gather information on the capabilities of the browser that is running on the client.

System.Web.HttpBrowserCapabilities properties are accessible through the **System.Web.HttpRequest.Browser** property of ASP.NET's intrinsic **System.Web.HttpContext.Request** object.

HttpBrowserCapabilities

Example Syntax:

UnLock

```
[C#] public HttpBrowserCapabilities();
[C++] public: HttpBrowserCapabilities();
[VB] Public Sub New()
[JScript] public function HttpBrowserCapabilities();
```

ActiveXControls

UnLock

[C#] public bool ActiveXControls {get;}
[C++] public: __property bool get_ActiveXControls();
[VB] Public ReadOnly Property ActiveXControls As Boolean
[JScript] public function get ActiveXControls() : Boolean;

Description

Gets a value indicating whether the client browser supports ActiveX controls.

AOL

UnLock

[C#] public bool AOL {get;}
[C++] public: __property bool get_AOL();
[VB] Public ReadOnly Property AOL As Boolean
[JScript] public function get AOL() : Boolean;

Description

Gets a value indicating whether the client is an America Online (AOL) browser.

BackgroundSounds

UnLock

[C#] public bool BackgroundSounds {get;}
[C++] public: __property bool get_BackgroundSounds();

```
[VB] Public ReadOnly Property BackgroundSounds As Boolean
[JScript] public function get BackgroundSounds() : Boolean;
```

Description

Gets a value indicating whether the client browser supports background sounds.

Beta

UnLock

```
[C#] public bool Beta {get;}
[C++] public: __property bool get_Beta();
[VB] Public ReadOnly Property Beta As Boolean
[JScript] public function get Beta() : Boolean;
```

Description

Gets a value indicating whether the browser is a beta release.

Browser

UnLock

```
[C#] public string Browser {get;}
[C++] public: __property String* get_Browser();
[VB] Public ReadOnly Property Browser As String
[JScript] public function get Browser() : String;
```

Description

Gets the browser string (if any) that was transmitted in the **User-Agent** header.

CDF

UnLock

[C#] public bool CDF {get;}
[C++] public: __property bool get_CDF();
[VB] Public ReadOnly Property CDF As Boolean
[JScript] public function get CDF() : Boolean;

Description

Gets a value indicating whether the client browser supports Channel Definition Format (CDF) for webcasting.

ClrVersion

UnLock

[C#] public Version ClrVersion {get;}
[C++] public: __property Version* get_ClrVersion();
[VB] Public ReadOnly Property ClrVersion As Version
[JScript] public function get ClrVersion() : Version;

Description

Gets the version number of the .NET common language runtime that the client browser supports.

If no common language runtime version is specified, the property value is 0, 0,-1,-1.

Cookies

UnLock

```
[C#] public bool Cookies {get;}
[C++] public: __property bool get_Cookies();
[VB] Public ReadOnly Property Cookies As Boolean
[JScript] public function get Cookies() : Boolean;
```

Description

Gets a value indicating whether the client browser supports cookies.

Crawler

UnLock

```
[C#] public bool Crawler {get;}
[C++] public: __property bool get_Crawler();
[VB] Public ReadOnly Property Crawler As Boolean
[JScript] public function get Crawler() : Boolean;
```

Description

Gets a value indicating whether the client browser is a Web crawler search engine.

EcmaScriptVersion

UnLock

[C#] public Version EcmaScriptVersion {get;}
[C++] public: __property Version* get_EcmaScriptVersion();
[VB] Public ReadOnly Property EcmaScriptVersion As Version
[JScript] public function get EcmaScriptVersion() : Version;

Description

Gets the version number of ECMA script that the client browser supports.

The European Computer Manufacturer's Association develops standards for information and communication systems. For more information, see ECMA's official Web site at http://www.ecma.ch.

Frames

UnLock

[C#] public bool Frames {get;}
[C++] public: __property bool get_Frames();
[VB] Public ReadOnly Property Frames As Boolean
[JScript] public function get Frames() : Boolean;

Description

Gets a value indicating whether the client browser supports HTML frames.

Item

JavaApplets

UnLock

Description

Gets a value indicating whether the client browser supports Java applets.

JavaScript

UnLock

[C#] public bool JavaScript {get;}
[C++] public: __property bool get_JavaScript();
[VB] Public ReadOnly Property JavaScript As Boolean
[JScript] public function get JavaScript() : Boolean;

Description

Gets a value indicating whether the client browser supports JavaScript.

MajorVersion

UnLock

[C#] public int MajorVersion {get;}
[C++] public: __property int get_MajorVersion();
[VB] Public ReadOnly Property MajorVersion As Integer
[JScript] public function get MajorVersion() : int;

Description

Gets the major (that is, integer) version number of the client browser.

MinorVersion

UnLock

[C#] public double MinorVersion {get;}
[C++] public: __property double get_MinorVersion();
[VB] Public ReadOnly Property MinorVersion As Double
[JScript] public function get MinorVersion() : double;

Description

Gets the minor (that is, decimal) version number of the client browser.

MSDomVersion

UnLock

[C#] public Version MSDomVersion {get;}
[C++] public: __property Version* get_MSDomVersion();
[VB] Public ReadOnly Property MSDomVersion As Version
[JScript] public function get MSDomVersion() : Version;

Description

Gets the version of Microsoft HTML (MSHTML) Document Object Model (DOM) that the client browser supports.

Platform

UnLock

[C#] public string Platform {get;}
[C++] public: __property String* get_Platform();

[VB] Public ReadOnly Property Platform As String
[JScript] public function get Platform() : String;

Description

Gets the name of the platform that the client uses.

Some possible **Platform** values are: Unknown, Win95, Win98, WinNT (which includes Windows 2000), Win16, WinCE, Mac68K, MacPPC, UNIX, and WebTV.

Tables

UnLock

[C#] public bool Tables {get;}
[C++] public: __property bool get_Tables();
[VB] Public ReadOnly Property Tables As Boolean
[JScript] public function get Tables() : Boolean;

Description

Gets a value indicating whether the client browser supports HTML tables.

TagWriter

UnLock

[C#] public Type TagWriter {get;}
[C++] public: __property Type* get_TagWriter();
[VB] Public ReadOnly Property TagWriter As Type
[JScript] public function get TagWriter() : Type;

Description

Type

UnLock

```
[C#] public string Type {get;}
[C++] public: __property String* get_Type();
[VB] Public ReadOnly Property Type As String
[JScript] public function get Type() : String;
```

Description

Gets the name and major (that is, integer) version number of the client browser.

VBScript

UnLock

```
[C#] public bool VBScript {get;}
[C++] public: __property bool get_VBScript();
[VB] Public ReadOnly Property VBScript As Boolean
[JScript] public function get VBScript() : Boolean;
```

Description

Gets a value indicating whether the client browser supports VBScript.

Version

UnLock

[C#] public string Version {get;}
[C++] public: __property String* get_Version();
[VB] Public ReadOnly Property Version As String
[JScript] public function get Version() : String;

Description

Gets the full (integer and decimal) version number of the client browser.

W3CDomVersion

UnLock

[C#] public Version W3CDomVersion {get;}
[C++] public: __property Version* get_W3CDomVersion();
[VB] Public ReadOnly Property W3CDomVersion As Version
[JScript] public function get W3CDomVersion() : Version;

Description

Gets the version of the World Wide Web Consortium (W3C) XML Document Object Model (DOM) that the client browser supports.

Win16

UnLock

[C#] public bool Win16 {get;}
[C++] public: __property bool get_Win16();

[VB] Public ReadOnly Property Win16 As Boolean
[JScript] public function get Win16() : Boolean;

Description

Gets a value indicating whether the client is a Win16-based machine.

Win32

UnLock

[C#] public bool Win32 {get;}
[C++] public: __property bool get_Win32();
[VB] Public ReadOnly Property Win32 As Boolean
[JScript] public function get Win32() : Boolean;

Description

Gets a value indicating whether the client is a Win32-based machine.

HttpCacheability enumeration (System.Web)

ToString

Description

Provides enumerated values that are used to set the **Cache-Control** HTTP header.

ToString

[C#] public const HttpCacheability NoCache;

```
[C++] public: const HttpCacheability NoCache;
[VB] Public Const NoCache As HttpCacheability
[JScript] public var NoCache : HttpCacheability;
```

Description

Sets the **Cache-Control: no-cache** header. Without a field name, the directive applies to the entire request and a shared (proxy server) cache must force a successful revalidation with the origin Web server before satisfying the request. With a field name, the directive applies only to the named field;the rest of the response may be supplied from a shared cache.

ToString

```
[C#] public const HttpCacheability Private;
[C++] public: const HttpCacheability Private;
[VB] Public Const Private As HttpCacheability
[JScript] public var Private : HttpCacheability;
```

Description

Default value. Sets **Cache-Control: private** to specify that the response is cacheable only on the client and not by shared (proxy server) caches.

ToString

```
[C#] public const HttpCacheability Public;
[C++] public: const HttpCacheability Public;
[VB] Public Const Public As HttpCacheability
```

[JScript] public var Public : HttpCacheability;

Description

Sets **Cache-Control: public** to specify that the response is cacheable by clients and shared (proxy) caches.

ToString

[C#] public const HttpCacheability Server;
[C++] public: const HttpCacheability Server;
[VB] Public Const Server As HttpCacheability
[JScript] public var Server : HttpCacheability;

Description

Specifies that the response is cached only at the origin server. Similar to the **NoCache** option. Clients receive a **Cache-Control: no-cache** directive but the document is cached on the origin server.

HttpCachePolicy class (System.Web)

ToString

Description

Contains methods for setting cache-specific HTTP headers and for controlling the ASP.NET page output cache.

For background information on HTTP headers and controlling caching, see the document RFC 2616: Hypertext Transfer Protocol - HTTP/1.1, available on the World Wide Web Consortium's site at http://www.w3c.org.

VaryByHeaders

ToString

```
[C#] public HttpCacheVaryByHeaders VaryByHeaders {get;}
[C++] public: __property HttpCacheVaryByHeaders* get_VaryByHeaders();
[VB] Public ReadOnly Property VaryByHeaders As HttpCacheVaryByHeaders
[JScript] public function get VaryByHeaders() : HttpCacheVaryByHeaders;
```

Description

Gets the list of all HTTP headers that will be used to vary cache output.

When a cached item has several vary headers, a separate version of the requested document is available from the cache for each HTTP header type.

VaryByParams

ToString

```
[C#] public HttpCacheVaryByParams VaryByParams {get;}
[C++] public: __property HttpCacheVaryByParams* get_VaryByParams();
[VB] Public ReadOnly Property VaryByParams As HttpCacheVaryByParams
[JScript] public function get VaryByParams() : HttpCacheVaryByParams;
```

Description

Gets the list of parameters received by a **GET** (querystring) or **POST** (in the body of the HTTP request) that affect caching.

For each named parameter in **VaryByParams** a separate version of the requested document is available from the cache, the version varying by the parameter's value.

AddValidationCallback

[C#] public void AddValidationCallback(HttpCacheValidateHandler handler, object data);

[C++] public: void AddValidationCallback(HttpCacheValidateHandler* handler, Object* data);

[VB] Public Sub AddValidationCallback(ByVal handler As HttpCacheValidateHandler, ByVal data As Object)

[JScript] public function AddValidationCallback(handler : HttpCacheValidateHandler, data : Object);

Description

Registers a validation callback for the current response.

AddValidationCallback provides a mechanism to programmatically check the validity of a item in the cache before the item is returned from the cache. The **System.Web.HttpCacheValidateHandler** value. The arbitrary user-supplied data that is passed back to the **AddValidationCallback** delegate.

AppendCacheExtension

[C#] public void AppendCacheExtension(string extension);

[C++] public: void AppendCacheExtension(String* extension);
[VB] Public Sub AppendCacheExtension(ByVal extension As String)
[JScript] public function AppendCacheExtension(extension : String);

Description

Appends the specified text to the **Cache-Control** HTTP header.

If the browser does not recognize cache control directives or extensions, the browser must ignore the unrecognized terms. For more information, see the document RFC 2616: Hypertext Transfer Protocol - HTTP/1.1, available on the World Wide Web Consortium's site at http://www.w3c.org . The text to append to the **Cache-Control** header.

SetCacheability

[C#] public void SetCacheability(HttpCacheability cacheability);
[C++] public: void SetCacheability(HttpCacheability cacheability);
[VB] Public Sub SetCacheability(ByVal cacheability As HttpCacheability)
[JScript] public function SetCacheability(cacheability : HttpCacheability); Sets the **Cache-Control** HTTP header. The **Cache-Control** HTTP header controls how documents are to be cached on the network.

Description

Sets the **Cache-Control** header to one of the values of **System.Web.HttpCacheability** . An **System.Web.HttpCacheability** enumeration value.

SetCacheability

[C#] public void SetCacheability(HttpCacheability cacheability, string field);
[C++] public: void SetCacheability(HttpCacheability cacheability, String* field);
[VB] Public Sub SetCacheability(ByVal cacheability As HttpCacheability, ByVal field As String)
[JScript] public function SetCacheability(cacheability : HttpCacheability, field : String);

Description

Sets the **Cache-Control** header to one of the values of **System.Web.HttpCacheability** and appends an extension to the directive.

The field name extension is valid only when used with the **private** or **no-cache** directives. For more information, see the document RFC 2616: Hypertext Transfer Protocol - HTTP/1.1, available on the World Wide Web Consortium's site at http://www.w3c.org. The **System.Web.HttpCacheability** enumeration value to set the header to. The cache control extension to add to the header.

SetETag

[C#] public void SetETag(string etag);
[C++] public: void SetETag(String* etag);
[VB] Public Sub SetETag(ByVal etag As String)
[JScript] public function SetETag(etag : String);

Description

Sets the **ETag** HTTP header to the specified string.

The ETag header is a unique identifier for a specific version of a document. Once an **ETag** header is set, subsequent attempts to set it will fail and an exception will be thrown. The text to use for the **ETag** header.

SetETagFromFileDependencies

```
[C#]        public     void       SetETagFromFileDependencies();
[C++]       public:    void       SetETagFromFileDependencies();
[VB]        Public     Sub        SetETagFromFileDependencies()
[JScript]   public     function   SetETagFromFileDependencies();
```

Description

Sets the **ETag** HTTP header based on the time stamps of the handler's file dependencies.

SetEtagFromFileDependencies sets the **ETag** header by retrieving the last modified time stamps of all files on which the handler is dependent, combining all file names and time stamps into a single string, then hashing that string into a single digest that is used as the **ETag** .

SetExpires

```
[C#]        public     void       SetExpires(DateTime          date);
[C++]       public:    void       SetExpires(DateTime          date);
[VB]    Public    Sub    SetExpires(ByVal    date    As    DateTime)
[JScript]    public    function    SetExpires(date    :    DateTime);
```

Description

Sets the **Expires** HTTP header to an absolute date and time.

This method will fail if the expiration date violates the principle of restrictiveness. The absolute **System.DateTime** value to set the **Expires**header to.

SetLastModified

[C#] public void SetLastModified(DateTime date);
[C++] public: void SetLastModified(DateTime date);
[VB] Public Sub SetLastModified(ByVal date As DateTime)
[JScript] public function SetLastModified(date : DateTime);

Description

Sets the **Last-Modified** HTTP header to the **System.DateTime** value supplied.

The **Last-Modified** HTTP header time stamps the document with the **DateTime** value indicating when the document was last modified. The new **System.DateTime** value for the **Last-Modified** header.

SetLastModifiedFromFileDependencies

[C#] public void SetLastModifiedFromFileDependencies();
[C++] public: void SetLastModifiedFromFileDependencies();
[VB] Public Sub SetLastModifiedFromFileDependencies()
[JScript] public function SetLastModifiedFromFileDependencies();

Description

Sets the **Last-Modified** HTTP header based on the time stamps of the handler's file dependencies.

SetMaxAge

```
[C#]        public      void      SetMaxAge(TimeSpan       delta);
[C++]       public:     void      SetMaxAge(TimeSpan       delta);
[VB]   Public   Sub   SetMaxAge(ByVal   delta   As   TimeSpan)
[JScript]   public   function   SetMaxAge(delta   :   TimeSpan);
```

Description

Sets the **Cache-Control: max-age** HTTP header based on the specified time span.

Max-age is the maximum absolute time a document is allowed to exist before being considered stale. The time span used to set the **Cache-Control: max-age** header.

SetNoServerCaching

```
[C#]          public       void        SetNoServerCaching();
[C++]         public:      void        SetNoServerCaching();
[VB]          Public       Sub         SetNoServerCaching()
[JScript]     public       function    SetNoServerCaching();
```

Description

Stops all origin-server caching for the current response.

Explicitly denies caching of the document on the origin-server. Once set, all requests for the document are fully processed. When this method is invoked, caching cannot be reenabled for the current response.

SetNoStore

```
[C#]          public      void        SetNoStore();
[C++]         public:     void        SetNoStore();
[VB]          Public      Sub         SetNoStore()
[JScript]     public      function    SetNoStore();
```

Description

Sets the **Cache-Control: no-store** directive.

SetNoTransforms

```
[C#]          public      void        SetNoTransforms();
[C++]         public:     void        SetNoTransforms();
[VB]          Public      Sub         SetNoTransforms()
[JScript]     public      function    SetNoTransforms();
```

Description

Sets the **CacheControl: no-transform** directive.

The **no-transform CacheControl** setting instructs network caching applications to not modify the document.

SetProxyMaxAge

```
[C#] public void SetProxyMaxAge(TimeSpan delta);
[C++] public: void SetProxyMaxAge(TimeSpan delta);
[VB] Public Sub SetProxyMaxAge(ByVal delta As TimeSpan)
[JScript] public function SetProxyMaxAge(delta : TimeSpan);
```

Description

Sets the **Cache-Control: s-maxage** HTTP header based on the specified time span.

System.Web.HttpCachePolicy.SetProxyMaxAge(System.TimeSpan) does not use sliding expiration and will fail if the expiration date violates the principle of restrictiveness. The time span used to set the **Cache-Control: s-maxage** header.

SetRevalidation

```
[C#] public void SetRevalidation(HttpCacheRevalidation revalidation);
[C++] public: void SetRevalidation(HttpCacheRevalidation revalidation);
[VB] Public Sub SetRevalidation(ByVal revalidation As HttpCacheRevalidation)
[JScript] public function SetRevalidation(revalidation : HttpCacheRevalidation);
```

Description

Sets the **Cache-Control** HTTP header to either the **must-revalidate** or the **proxy-revalidate** directives based on the supplied enumeration value.

The default is to send neither directive in a header unless explicitly specified by this method. The **System.Web.HttpCacheRevalidation** enumeration value to set the **Cache-Control** header to.

SetSlidingExpiration

```
[C#] public void SetSlidingExpiration(bool slide);
[C++] public: void SetSlidingExpiration(bool slide);
[VB] Public Sub SetSlidingExpiration(ByVal slide As Boolean)
[JScript] public function SetSlidingExpiration(slide : Boolean);
```

Description

Sets cache expiration to sliding.

When cache expiration is set to sliding, the **Cache-Control** HTTP header will be renewed with each response. This expiration mode is identical to the IIS configuration option to add an expiration header to all output set relative to the current time. **true** or **false** .

SetValidUntilExpires

```
[C#] public void SetValidUntilExpires(bool validUntilExpires);
[C++] public: void SetValidUntilExpires(bool validUntilExpires);
[VB] Public Sub SetValidUntilExpires(ByVal validUntilExpires As Boolean)
[JScript] public function SetValidUntilExpires(validUntilExpires : Boolean);
```

Description

SetVaryByCustom

```
[C#] public void SetVaryByCustom(string custom);
[C++] public: void SetVaryByCustom(String* custom);
[VB] Public Sub SetVaryByCustom(ByVal custom As String)
[JScript] public function SetVaryByCustom(custom : String);
```

Description

Sets the **Vary** HTTP header to the specified text string. The text to set the **Vary** header to.

HttpCacheRevalidation enumeration (System.Web)

ToString

Description

Provides enumerated values that are used to set revalidation-specific **Cache-Control** HTTP headers.

ToString

```
[C#] public const HttpCacheRevalidation AllCaches;
[C++] public: const HttpCacheRevalidation AllCaches;
[VB] Public Const AllCaches As HttpCacheRevalidation
[JScript] public var AllCaches : HttpCacheRevalidation;
```

Description

Sets the **Cache-Control: must-revalidate** HTTP header.

ToString

[C#] public const HttpCacheRevalidation None;

[C++] public: const HttpCacheRevalidation None;

[VB] Public Const None As HttpCacheRevalidation

[JScript] public var None : HttpCacheRevalidation;

Description

Default value. If this value is set, no cache-revalidation directive is sent.

ToString

[C#] public const HttpCacheRevalidation ProxyCaches;

[C++] public: const HttpCacheRevalidation ProxyCaches;

[VB] Public Const ProxyCaches As HttpCacheRevalidation

[JScript] public var ProxyCaches : HttpCacheRevalidation;

Description

Sets the **Cache-Control: proxy-revalidate** HTTP header.

HttpCacheValidateHandler delegate (System.Web)

ToString

Description

Delegate method that is called when a cached item is validated. Cache items invalidated within the method are treated as cache misses. The **System.Web.HttpContext** object containing information about the current request. User-supplied data used to validate the cached item. A **System.Web.HttpValidationStatus** enumeration value.

If any handler invalidates the cached item, the item is evicted from the cache and the request is handled as a cache miss.

HttpCacheVaryByHeaders class (System.Web)

ToString

Description

Provides a type-safe way to set the **Vary** HTTP header.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request.

AcceptTypes

ToString

[C#] public bool AcceptTypes {get; set;}
[C++] public: __property bool get_AcceptTypes();public: __property void set_AcceptTypes(bool);
[VB] Public Property AcceptTypes As Boolean
[JScript] public function get AcceptTypes() : Boolean;public function set AcceptTypes(Boolean);

Description

Gets or sets a value indicating whether the origin server adds the **Accept** field to the **Vary** HTTP header.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request. The **Accept** field specifies that the server selects the response based on the media types acceptable to the client.

Item

ToString

[C#] public bool this[string header] {get; set;}
[C++] public: __property bool get_Item(String* header);public: __property void set_Item(String* header, bool);
[VB] Public Default Property Item(ByVal header As String) As Boolean
[JScript] returnValue = HttpCacheVaryByHeadersObject.Item(header);HttpCacheVaryByHeadersObject.Item(header) = returnValue;

Description

Gets or sets a value indicating whether the origin server should add a custom field to the **Vary** HTTP header.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request. The name of the custom header.

UserAgent

ToString

[C#] public bool UserAgent {get; set;}
[C++] public: __property bool get_UserAgent();public: __property void set_UserAgent(bool);
[VB] Public Property UserAgent As Boolean
[JScript] public function get UserAgent() : Boolean;public function set UserAgent(Boolean);

Description

Gets or sets a value indicating whether the origin server adds the **User-Agent** field to the **Vary** HTTP header.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request. The **User-Agent** field specifies that the server selects the response based on the type of client user-agent.

UserCharSet

ToString

[C#] public bool UserCharSet {get; set;}
[C++] public: __property bool get_UserCharSet();public: __property void set_UserCharSet(bool);
[VB] Public Property UserCharSet As Boolean
[JScript] public function get UserCharSet() : Boolean;public function set

UserCharSet(Boolean);

Description

Gets or sets a value indicating whether the origin server should add the **Accept-Charset** field to the **Vary** HTTP header.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request. The **Accept-CharSet** field specifies that the server selects the response based on the client's character set.

UserLanguage

ToString

[C#] public bool UserLanguage {get; set;}
[C++] public: __property bool get_UserLanguage();public: __property void set_UserLanguage(bool);
[VB] Public Property UserLanguage As Boolean
[JScript] public function get UserLanguage() : Boolean;public function set UserLanguage(Boolean);

Description

Gets or sets a value indicating whether the origin server adds the **Accept-Language** field to the **Vary** HTTP header.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request. The **Accept-Language** field specifies that the server selects the response based on languages acceptable to the client.

VaryByUnspecifiedParameters

[C#]	public	void	VaryByUnspecifiedParameters();
[C++]	public:	void	VaryByUnspecifiedParameters();
[VB]	Public	Sub	VaryByUnspecifiedParameters()
[JScript]	public	function	VaryByUnspecifiedParameters();

Description

Sets the **Vary** HTTP header to the value * (an asterisk) and causes all other **Vary** header information to be dropped.

The **Vary** header indicates the request-header fields that the server uses to determine which of multiple cached responses is sent in response to a client request. The * field specifies that the server selects the response based on parameters not specified in request headers (for example, the network address of the client).

HttpCacheVaryByParams class (System.Web)

VaryByUnspecifiedParameters

Description

Indicates that a cache should contain multiple representations (cached responses) for a particular URI. This class is an encapsulation that provides a type-safe way to set the **Vary** HTTP header.

For more information on HTTP cache control headers, see RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1, available on the World Wide Web Consortium's Web site at http://www.w3c.org. See section 14, "Header Field Definitions", for complete details.

IgnoreParams

VaryByUnspecifiedParameters

[C#] public bool IgnoreParams {get; set;}
[C++] public: __property bool get_IgnoreParams();public: __property void set_IgnoreParams(bool);
[VB] Public Property IgnoreParams As Boolean
[JScript] public function get IgnoreParams() : Boolean;public function set IgnoreParams(Boolean);

Description

Gets or sets a value indicating whether HTTP header cache control parameters are ignored.

Item

VaryByUnspecifiedParameters

[C#] public bool this[string header] {get; set;}
[C++] public: __property bool get_Item(String* header);public: __property void set_Item(String* header, bool);
[VB] Public Default Property Item(ByVal header As String) As Boolean
[JScript] returnValue =

HttpCacheVaryByParamsObject.Item(header);HttpCacheVaryByParamsObject.Item(header) = returnValue;

Description

Gets or sets the name of the cache-control header that is used to select one of several different cached responses. The name of the custom header.

HttpClientCertificate class (System.Web)

ToString

Description

The **HttpClientCertificate** collection retrieves the certification fields (specified in the X.509 standard) from a request issued by the Web browser.

AllKeys

BinaryIssuer

ToString

Description

CertEncoding

ToString

```
[C#]      public      int        CertEncoding       {get;}
[C++]     public:     __property int     get_CertEncoding();
```

[VB] Public ReadOnly Property CertEncoding As Integer
[JScript] public function get CertEncoding() : int;

Description

Certificate

ToString

[C#] public byte[] Certificate {get;}
[C++] public: __property unsigned char get_Certificate();
[VB] Public ReadOnly Property Certificate As Byte ()
[JScript] public function get Certificate() : Byte[];

Description

A string containing the binary stream of the entire certificate content in ASN.1 format.

Cookie

ToString

[C#] public string Cookie {get;}
[C++] public: __property String* get_Cookie();
[VB] Public ReadOnly Property Cookie As String
[JScript] public function get Cookie() : String;

Description

Count

Flags

ToString

Description

A set of flags that provide additional client certificate information.

IsPresent

ToString

```
[C#]        public         bool          IsPresent           {get;}
[C++]       public:        __property    bool                get_IsPresent();
[VB]        Public    ReadOnly    Property    IsPresent    As    Boolean
[JScript]   public    function    get    IsPresent()    :    Boolean;
```

Description

IsReadOnly

Issuer

ToString

Description

A string that contains a list of subfield values containing information about the certificate issuer.

IsValid

ToString

```
[C#]        public      bool        IsValid       {get;}
[C++]       public:     __property  bool          get_IsValid();
[VB]        Public      ReadOnly    Property      IsValid   As   Boolean
[JScript]   public      function    get           IsValid()  :   Boolean;
```

Description

Item

Item

Keys

KeySize

ToString

Description

PublicKey

ToString

```
[C#]        public      byte[]      PublicKey     {get;}
```

[C++] public: __property unsigned char get_PublicKey();
[VB] Public ReadOnly Property PublicKey As Byte ()
[JScript] public function get PublicKey() : Byte[];

Description

SecretKeySize

ToString

[C#] public int SecretKeySize {get;}
[C++] public: __property int get_SecretKeySize();
[VB] Public ReadOnly Property SecretKeySize As Integer
[JScript] public function get SecretKeySize() : int;

Description

SerialNumber

ToString

[C#] public string SerialNumber {get;}
[C++] public: __property String* get_SerialNumber();
[VB] Public ReadOnly Property SerialNumber As String
[JScript] public function get SerialNumber() : String;

Description

A string that contains the certification serial number as an ASCII representation of hexadecimal bytes separated by hyphens (-). For example, 04-67-F3-02.

ServerIssuer

ToString

```
[C#]        public        string        ServerIssuer        {get;}
[C++]       public:       __property    String*     get_ServerIssuer();
[VB]    Public    ReadOnly    Property    ServerIssuer    As    String
[JScript]   public   function   get   ServerIssuer()   :   String;
```

Description

ServerSubject

ToString

```
[C#]        public        string        ServerSubject        {get;}
[C++]       public:       __property    String*     get_ServerSubject();
[VB]    Public    ReadOnly    Property    ServerSubject    As    String
[JScript]   public   function   get   ServerSubject()   :   String;
```

Description

Subject

ToString

```
[C#]        public        string        Subject        {get;}
[C++]       public:       __property    String*        get_Subject();
[VB]        Public        ReadOnly      Property       Subject      As      String
[JScript]   public        function      get            Subject()    :       String;
```

Description

A string that contains a list of subfield values that contain information about the subject of the certificate. If this value is specified without a *SubField* , the ClientCertificate collection returns a comma-separated list of subfields. For example, C=US, O=Msft, and so on.

ValidFrom

ToString

```
[C#]        public        DateTime      ValidFrom      {get;}
[C++]       public:       __property    DateTime       get_ValidFrom();
[VB]        Public        ReadOnly      Property       ValidFrom    As      DateTime
[JScript]   public        function      get            ValidFrom()  :       DateTime;
```

Description

A date specifying when the certificate becomes valid. This date varies with international settings.

ValidUntil

ToString

[C#] public DateTime ValidUntil {get;}

[C++] public: __property DateTime get_ValidUntil();

[VB] Public ReadOnly Property ValidUntil As DateTime

[JScript] public function get ValidUntil() : DateTime;

Description

A date specifying when the certificate expires. The year value is displayed as a four-digit number.

Get

[C#] public override string Get(string field);

[C++] public: String* Get(String* field);

[VB] Overrides Public Function Get(ByVal field As String) As String

[JScript] public override function Get(field : String) : String;

Description

Allows access to individual items in the collection by name. The name of the item in the collection to retrieve.

HttpCompileException class (System.Web)

ToString

Description

The exception that is thrown when a compiler error occurs.

HttpCompileException

Example Syntax:

ToString

[C#] public HttpCompileException(CompilerResults results, string sourceCode);

[C++] public: HttpCompileException(CompilerResults* results, String* sourceCode);

[VB] Public Sub New(ByVal results As CompilerResults, ByVal sourceCode As String)

[JScript] public function HttpCompileException(results : CompilerResults, sourceCode : String);

Description

Initializes a new instance of the **System.Web.HttpCompileException** class. A **System.CodeDom.Compiler.CompilerResults** containing compiler output and error information. The name of the file being compiled when the error occurs.

ErrorCode

HelpLink

HResult

InnerException

Message

Results

ToString

Description

Gets compiler output and error information for the exception.

Source

SourceCode

ToString

Description

Gets the name of the source file being compiled when the error occurs.

StackTrace

TargetSite

HttpContext class (System.Web)

ToString

Description

Encapsulates all HTTP-specific information about an individual HTTP request.

Classes that inherit the **System.Web.IHttpModule** and **System.Web.IHttpHandler** interfaces are provided a reference to an **HttpContext** object for the current HTTP request. The object provides access to the intrinsic **System.Web.HttpContext.Request** ,

System.Web.HttpContext.Response , and **System.Web.HttpContext.Server** objects for the request.

HttpContext

Example Syntax:

ToString

[C#] public HttpContext(HttpWorkerRequest wr);
[C++] public: HttpContext(HttpWorkerRequest* wr);
[VB] Public Sub New(ByVal wr As HttpWorkerRequest)
[JScript] public function HttpContext(wr : HttpWorkerRequest);

Description

Initializes a new instance of the **System.Web.HttpContext** class. The **System.Web.HttpWorkerRequest** object for the current HTTP request.

HttpContext

Example Syntax:

ToString

[C#] public HttpContext(HttpRequest request, HttpResponse response);
[C++] public: HttpContext(HttpRequest* request, HttpResponse* response);
[VB] Public Sub New(ByVal request As HttpRequest, ByVal response As HttpResponse)
[JScript] public function HttpContext(request : HttpRequest, response : HttpResponse); Initializes a new instance of the **System.Web.HttpContext** class.

Description

Initializes a new instance of the **System.Web.HttpContext** class. The **System.Web.HttpRequest** object for the current HTTP request. The **System.Web.HttpResponse** object for the current HTTP request.

AllErrors

ToString

[C#] public Exception[] AllErrors {get;}
[C++] public: __property Exception* get_AllErrors();
[VB] Public ReadOnly Property AllErrors As Exception ()
[JScript] public function get AllErrors() : Exception[];

Description

Gets an array of errors accumulated while processing an HTTP request.

Application

ToString

[C#] public HttpApplicationState Application {get;}
[C++] public: __property HttpApplicationState* get_Application();
[VB] Public ReadOnly Property Application As HttpApplicationState
[JScript] public function get Application() : HttpApplicationState;

Description

Gets the **System.Web.HttpApplicationState** object for the current HTTP request.

ApplicationInstance

ToString

[C#] public HttpApplication ApplicationInstance {get; set;}
[C++] public: __property HttpApplication* get_ApplicationInstance();public: __property void set_ApplicationInstance(HttpApplication*);
[VB] Public Property ApplicationInstance As HttpApplication
[JScript] public function get ApplicationInstance() : HttpApplication;public function set ApplicationInstance(HttpApplication);

Description

Gets or sets the **System.Web.HttpApplicationState** object for the current HTTP request.

Cache

ToString

[C#] public Cache Cache {get;}
[C++] public: __property Cache* get_Cache();
[VB] Public ReadOnly Property Cache As Cache
[JScript] public function get Cache() : Cache;

Description

Gets the **System.Web.Caching.Cache** object for the current HTTP request.

Current

ToString

[C#] public static HttpContext Current {get;}
[C++] public: __property static HttpContext* get_Current();
[VB] Public Shared ReadOnly Property Current As HttpContext
[JScript] public static function get Current() : HttpContext;

Description

Gets the **System.Web.HttpContext** object for the current HTTP request.

Error

ToString

[C#] public Exception Error {get;}
[C++] public: __property Exception* get_Error();
[VB] Public ReadOnly Property Error As Exception
[JScript] public function get Error() : Exception;

Description

Gets the first error (if any) accumulated during HTTP request processing.

Handler

ToString

[C#] public IHttpHandler Handler {get; set;}
[C++] public: __property IHttpHandler* get_Handler();public: __property void

set_Handler(IHttpHandler*);
[VB] Public Property Handler As IHttpHandler
[JScript] public function get Handler() : IHttpHandler;public function set Handler(IHttpHandler);

Description

Gets or sets the **System.Web.IHttpHandler** object for the current HTTP request.

IsCustomErrorEnabled

ToString

[C#] public bool IsCustomErrorEnabled {get;}
[C++] public: __property bool get_IsCustomErrorEnabled();
[VB] Public ReadOnly Property IsCustomErrorEnabled As Boolean
[JScript] public function get IsCustomErrorEnabled() : Boolean;

Description

Gets a value indicating whether custom errors are enabled for the current HTTP request.

IsDebuggingEnabled

ToString

[C#] public bool IsDebuggingEnabled {get;}
[C++] public: __property bool get_IsDebuggingEnabled();
[VB] Public ReadOnly Property IsDebuggingEnabled As Boolean

[JScript] public function get IsDebuggingEnabled() : Boolean;

Description

Gets a value indicating whether the current HTTP request is in debug mode.

Items

ToString

[C#] public IDictionary Items {get;}
[C++] public: __property IDictionary* get_Items();
[VB] Public ReadOnly Property Items As IDictionary
[JScript] public function get Items() : IDictionary;

Description

Gets a key-value collection that can be used to organize and share data between an **System.Web.IHttpModule** and an **System.Web.IHttpHandler** during an HTTP request.

Request

ToString

[C#] public HttpRequest Request {get;}
[C++] public: __property HttpRequest* get_Request();
[VB] Public ReadOnly Property Request As HttpRequest
[JScript] public function get Request() : HttpRequest;

Description

Gets the **System.Web.HttpRequest** object for the current HTTP request.

Response

ToString

[C#] public HttpResponse Response {get;}
[C++] public: __property HttpResponse* get_Response();
[VB] Public ReadOnly Property Response As HttpResponse
[JScript] public function get Response() : HttpResponse;

Description

Gets the **System.Web.HttpResponse** object for the current HTTP response.

Server

ToString

[C#] public HttpServerUtility Server {get;}
[C++] public: __property HttpServerUtility* get_Server();
[VB] Public ReadOnly Property Server As HttpServerUtility
[JScript] public function get Server() : HttpServerUtility;

Description

Gets the **System.Web.HttpServerUtility** object that provides methods used in processing Web requests.

Session

ToString

[C#] public HttpSessionState Session {get;}
[C++] public: __property HttpSessionState* get_Session();
[VB] Public ReadOnly Property Session As HttpSessionState
[JScript] public function get Session() : HttpSessionState;

Description

Gets the **System.Web.SessionState** instance for the current HTTP request.

SkipAuthorization

ToString

[C#] public bool SkipAuthorization {get; set;}
[C++] public: __property bool get_SkipAuthorization();public: __property void set_SkipAuthorization(bool);
[VB] Public Property SkipAuthorization As Boolean
[JScript] public function get SkipAuthorization() : Boolean;public function set SkipAuthorization(Boolean);

Description

Gets or sets a value that specifies whether the URLAuthorization module will skip the authorization check for the current request.

SkipAuthorization is for advanced use by authentication modules that need to redirect to an anonymous-allowed page. The Forms authentication module and Passport authentication module both set this property when redirecting to a configured login page. Setting this requires the **ControlPrincipal** flag to be set in **System.Security.Permissions.SecurityPermission.Flags** .

Timestamp

ToString

[C#] public DateTime Timestamp {get;}
[C++] public: __property DateTime get_Timestamp();
[VB] Public ReadOnly Property Timestamp As DateTime
[JScript] public function get Timestamp() : DateTime;

Description

Gets the initial timestamp of the current HTTP request.

Trace

ToString

[C#] public TraceContext Trace {get;}
[C++] public: __property TraceContext* get_Trace();
[VB] Public ReadOnly Property Trace As TraceContext
[JScript] public function get Trace() : TraceContext;

Description

Gets the **System.Web.TraceContext** object for the current HTTP response.

User

ToString

[C#] public IPrincipal User {get; set;}
[C++] public: __property IPrincipal* get_User();public: __property void set_User(IPrincipal*);
[VB] Public Property User As IPrincipal
[JScript] public function get User() : IPrincipal;public function set User(IPrincipal);

Description

Gets or sets security information for the current HTTP request.

Setting this property requires the **ControlPrincipal** flag to be set in **System.Security.Permissions.SecurityPermission.Flags** .

AddError

[C#] public void AddError(Exception errorInfo);
[C++] public: void AddError(Exception* errorInfo);
[VB] Public Sub AddError(ByVal errorInfo As Exception)
[JScript] public function AddError(errorInfo : Exception);

Description

Adds an exception to the exception collection for the current HTTP request. The **System.Exception** object to add to the exception collection.

ClearError

[C#] public void ClearError();

[C++] public: void ClearError();
[VB] Public Sub ClearError()
[JScript] public function ClearError();

Description

Clears all errors for the current HTTP request.

GetAppConfig

[C#] public static object GetAppConfig(string name);
[C++] public: static Object* GetAppConfig(String* name);
[VB] Public Shared Function GetAppConfig(ByVal name As String) As Object
[JScript] public static function GetAppConfig(name : String) : Object;

Description

Returns requested configuration information for the current application The application configuration tag that information is requested for.

GetConfig

[C#] public object GetConfig(string name);
[C++] public: Object* GetConfig(String* name);
[VB] Public Function GetConfig(ByVal name As String) As Object
[JScript] public function GetConfig(name : String) : Object; Returns requested configuration information for the current HTTP request.

Description

Returns requested configuration information for the current HTTP request. The configuration tag that information is requested for.

RewritePath

[C#] public void RewritePath(string path);
[C++] public: void RewritePath(String* path);
[VB] Public Sub RewritePath(ByVal path As String)
[JScript] public function RewritePath(path : String);

Description

Assigns an internal rewrite path. The internal rewrite path.

IServiceProvider.GetService

[C#] object IServiceProvider.GetService(Type service);
[C++] Object* IServiceProvider::GetService(Type* service);
[VB] Function GetService(ByVal service As Type) As Object Implements IServiceProvider.GetService
[JScript] function IServiceProvider.GetService(service : Type) : Object;

HttpCookie class (System.Web)

ToString

Description

Provides a type-safe way to create and manipulate individual HTTP cookies.

The **System.Web.HttpCookie** class gets and sets properties of individual cookies. The **System.Web.HttpCookieCollection** class provides methods to store, retrieve, and manage all the cookies for an entire Web application. ASP.NET code uses the intrinsic **System.Web.HttpResponse.Cookies** object to create cookies and add them to the cookie collection. When delivering a Web page to a client, the server sends the entire cookie collection with the **Set-Cookie** header.

HttpCookie

Example Syntax:

ToString

[C#] public HttpCookie(string name);
[C++] public: HttpCookie(String* name);
[VB] Public Sub New(ByVal name As String)
[JScript] public function HttpCookie(name : String); Initializes a new instance of the **System.Web.HttpCookie** class.

Description

Creates and names a new cookie. The name of the new cookie.

HttpCookie

Example Syntax:

ToString

[C#] public HttpCookie(string name, string value);
[C++] public: HttpCookie(String* name, String* value);

[VB] Public Sub New(ByVal name As String, ByVal value As String)
[JScript] public function HttpCookie(name : String, value : String);

Description

Creates, names, and assigns a value to a new cookie. The name of the new cookie. The value of the new cookie.

Domain

ToString

[C#] public string Domain {get; set;}
[C++] public: __property String* get_Domain();public: __property void set_Domain(String*);
[VB] Public Property Domain As String
[JScript] public function get Domain() : String;public function set Domain(String);

Description

Gets or sets the domain to associate the cookie with.

Setting the **Domain** attribute limits transmission of the cookie to clients requesting a resource from that domain.

Expires

ToString

[C#] public DateTime Expires {get; set;}
[C++] public: __property DateTime get_Expires();public: __property void set_Expires(DateTime);

[VB] Public Property Expires As DateTime
[JScript] public function get Expires() : DateTime;public function set Expires(DateTime);

Description

Gets or sets the expiration date and time for the cookie.

HasKeys

ToString

[C#] public bool HasKeys {get;}
[C++] public: __property bool get_HasKeys();
[VB] Public ReadOnly Property HasKeys As Boolean
[JScript] public function get HasKeys() : Boolean;

Description

Gets a value indicating whether a cookie has subkeys.

Item

ToString

[C#] public string this[string key] {get; set;}
[C++] public: __property String* get_Item(String* key);public: __property void set_Item(String* key, String*);
[VB] Public Default Property Item(ByVal key As String) As String
[JScript] returnValue = HttpCookieObject.Item(key);HttpCookieObject.Item(key) = returnValue;

Description

Shortcut for **HttpCookie.Values[** key **].** This property is provided for compatibility with previous versions of ASP. Key (index) of cookie value.

Name

ToString

[C#] public string Name {get; set;}

[C++] public: __property String* get_Name();public: __property void set_Name(String*);

[VB] Public Property Name As String

[JScript] public function get Name() : String;public function set Name(String);

Description

Gets or sets the name of a cookie.

Path

ToString

[C#] public string Path {get; set;}

[C++] public: __property String* get_Path();public: __property void set_Path(String*);

[VB] Public Property Path As String

[JScript] public function get Path() : String;public function set Path(String);

Description

Gets or sets the virtual path to transmit with the current cookie.

The **Path** property extends the **Domain** property to completely describe the specific URL that the cookie applies to. For example, in the URL http:/www.microsoft.com/asp, the domain is www.microsoft.com and the path is /asp.

Secure

ToString

[C#] public bool Secure {get; set;}
[C++] public: __property bool get_Secure();public: __property void set_Secure(bool);
[VB] Public Property Secure As Boolean
[JScript] public function get Secure() : Boolean;public function set Secure(Boolean);

Description

Gets or sets a value indicating whether to transmit the cookie securely (that is, over HTTPS only).

Value

ToString

[C#] public string Value {get; set;}
[C++] public: __property String* get_Value();public: __property void set_Value(String*);
[VB] Public Property Value As String

[JScript] public function get Value() : String;public function set Value(String);

Description

Gets or sets an individual cookie value.

Values

ToString

[C#] public NameValueCollection Values {get;}
[C++] public: __property NameValueCollection* get_Values();
[VB] Public ReadOnly Property Values As NameValueCollection
[JScript] public function get Values() : NameValueCollection;

Description

Gets a collection of key-and-value value pairs that are contained within a single cookie object.

HttpCookieCollection class (System.Web)

ToString

Description

Provides a type-safe way to manipulate HTTP cookies.

HttpCookieCollection

Example Syntax:

ToString

[C#] public HttpCookieCollection();
[C++] public: HttpCookieCollection();
[VB] Public Sub New()
[JScript] public function HttpCookieCollection();

Description

Initializes a new instance of the **System.Web.HttpCookieCollection** class.

ASP.NET includes two intrinsic cookie collections. The collection accessible through **System.Web.HttpRequest.Cookies** contains cookies transmitted by the client to the server in the **Cookie** header. The collection accessible through **System.Web.HttpResponse.Cookies** contains cookies generated on the server and transmitted to the client in the **Set-Cookie** header.

AllKeys

ToString

[C#] public string[] AllKeys {get;}
[C++] public: __property String* get_AllKeys();
[VB] Public ReadOnly Property AllKeys As String ()
[JScript] public function get AllKeys() : String[];

Description

Gets a string array containing all the keys (cookie names) in the cookie collection.

Count

IsReadOnly

Item

ToString

Description

Gets the cookie with the specified numerical index from the cookie collection. The index of the cookie to retrieve from the collection.

Item

ToString

[C#] public HttpCookie this[string name] {get;}
[C++] public: __property HttpCookie* get_Item(String* name);
[VB] Public Default ReadOnly Property Item(ByVal name As String) As HttpCookie
[JScript] returnValue = HttpCookieCollectionObject.Item(name); Gets the cookie with the specified name from the cookie collection. This property is overloaded to allow retrieval of cookies by either name or numerical index.

Description

Gets the cookie with the specified name from the cookie collection. Name of cookie to retrieve.

Keys

Add

[C#] public void Add(HttpCookie cookie);
[C++] public: void Add(HttpCookie* cookie);
[VB] Public Sub Add(ByVal cookie As HttpCookie)
[JScript] public function Add(cookie : HttpCookie);

Description

Adds the specified cookie to the cookie collection.

Any number of cookie collections can exist within an application, but only the collection referenced by the intrinsic **System.Web.HttpResponse.Cookies** object is sent to the client. The **System.Web.HttpCookie** to add to the collection.

Clear

[C#] public void Clear();
[C++] public: void Clear();
[VB] Public Sub Clear()
[JScript] public function Clear();

Description

Clears all cookies from the cookie collection.

CopyTo

[C#] public void CopyTo(Array dest, int index);
[C++] public: void CopyTo(Array* dest, int index);
[VB] Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)

[JScript] public function CopyTo(dest : Array, index : int);

Description

Copies members of the cookie collection to an **System.Array** beginning at the specified index of the array. The destination **System.Array**. The index of the destination array where copying starts.

Get

[C#] public HttpCookie Get(int index);
[C++] public: HttpCookie* Get(int index);
[VB] Public Function Get(ByVal index As Integer) As HttpCookie
[JScript] public function Get(index : int) : HttpCookie;

Description

Returns the **System.Web.HttpCookie** item with the specified index from the cookie collection. The index of the cookie to return from the collection.

Get

[C#] public HttpCookie Get(string name);
[C++] public: HttpCookie* Get(String* name);
[VB] Public Function Get(ByVal name As String) As HttpCookie
[JScript] public function Get(name : String) : HttpCookie; Returns an individual **System.Web.HttpCookie** object from the cookie collection. This property is overloaded to allow retrieval of cookies by either name or numerical index.

Description

Returns the **System.Web.HttpCookie** item with the specified name from the cookie collection.

If the named cookie does not exist, this method creates a new cookie with that name. The name of the cookie to retrieve from the collection.

GetKey

```
[C#] public string GetKey(int index);
[C++] public: String* GetKey(int index);
[VB] Public Function GetKey(ByVal index As Integer) As String
[JScript] public function GetKey(index : int) : String;
```

Description

Returns the key (name) of the cookie at the specified numerical index. The index of the key to retrieve from the collection.

Remove

```
[C#] public void Remove(string name);
[C++] public: void Remove(String* name);
[VB] Public Sub Remove(ByVal name As String)
[JScript] public function Remove(name : String);
```

Description

Removes the cookie with the specified name from the collection. The name of the cookie to remove from the collection.

Set

[C#] public void Set(HttpCookie cookie);
[C++] public: void Set(HttpCookie* cookie);
[VB] Public Sub Set(ByVal cookie As HttpCookie)
[JScript] public function Set(cookie : HttpCookie);

Description

Updates the value of an existing cookie in a cookie collection. The **System.Web.HttpCookie** object to update.

HttpException class (System.Web)

ToString

Description

Provides a means of generating HTTP exceptions.

HttpException

Example Syntax:

ToString

[C#] public HttpException();
[C++] public: HttpException();
[VB] Public Sub New()

[JScript] public function HttpException(); Constructs a new **System.Exception** object.

Description

Constructs an empty **Exception** object.

When handling exceptions, it is sometimes useful to capture a series of related exceptions with the outer exceptions being thrown in response to an inner exceptions.

HttpException

Example Syntax:

ToString

[C#] public HttpException(string message);
[C++] public: HttpException(String* message);
[VB] Public Sub New(ByVal message As String)
[JScript] public function HttpException(message : String);

Description

Constructs an **System.Exception** using a supplied error message. The message displayed to the client when the exception is thrown.

HttpException

Example Syntax:

ToString

[C#] public HttpException(int httpCode, string message);

[C++] public: HttpException(int httpCode, String* message);
[VB] Public Sub New(ByVal httpCode As Integer, ByVal message As String)
[JScript] public function HttpException(httpCode : int, message : String);

Description

Constructs an **System.Exception** using an HTTP error code and an error message. The HTTP error code displayed on the client. The message displayed to the client when the exception is thrown.

HttpException

Example Syntax:

ToString

[C#] public HttpException(string message, Exception innerException);
[C++] public: HttpException(String* message, Exception* innerException);
[VB] Public Sub New(ByVal message As String, ByVal innerException As Exception)
[JScript] public function HttpException(message : String, innerException : Exception);

Description

Constructs an **System.Exception** using an error message and the **System.Exception.InnerException** property.

When handling exceptions, it is sometimes useful to capture a series of related exceptions with the outer exceptions being thrown in response to an inner exception. The message displayed to the client when the exception is thrown. The **System.Exception.InnerException**, if any, that threw the current exception.

HttpException

Example Syntax:

ToString

[C#] public HttpException(string message, int hr);
[C++] public: HttpException(String* message, int hr);
[VB] Public Sub New(ByVal message As String, ByVal hr As Integer)
[JScript] public function HttpException(message : String, hr : int);

Description

Constructs an **System.Exception** using error message and an exception code. The error message displayed to the client when the exception is thrown. The exception code that defines the error.

HttpException

Example Syntax:

ToString

[C#] public HttpException(int httpCode, string message, Exception innerException);
[C++] public: HttpException(int httpCode, String* message, Exception* innerException);
[VB] Public Sub New(ByVal httpCode As Integer, ByVal message As String, ByVal innerException As Exception)

[JScript] public function HttpException(httpCode : int, message : String, innerException : Exception);

Description

Constructs an **System.Exception** using an HTTP error code, an error message, and the **System.Exception.InnerException** property.

When handling exceptions, it is sometimes useful to capture a series of related exceptions with the outer exceptions being thrown in response to an inner exceptions. The HTTP error code displayed to the client. The message displayed to the client. The **InnerException** , if any, that threw the current exception.

HttpException

Example Syntax:

ToString

[C#] public HttpException(int httpCode, string message, int hr);
[C++] public: HttpException(int httpCode, String* message, int hr);
[VB] Public Sub New(ByVal httpCode As Integer, ByVal message As String, ByVal hr As Integer)
[JScript] public function HttpException(httpCode : int, message : String, hr : int);

Description

Constructs an **System.Exception** using HTTP error code, an error message, and an exception code. The HTTP error code displayed on the client. The error message displayed to the client. The error code that defines the error.

ErrorCode

HelpLink

HResult

InnerException

Message

Source

StackTrace

TargetSite

CreateFromLastError

[C#] public static HttpException CreateFromLastError(string message);
[C++] public: static HttpException* CreateFromLastError(String* message);
[VB] Public Shared Function CreateFromLastError(ByVal message As String) As HttpException
[JScript] public static function CreateFromLastError(message : String) : HttpException;

Description

Creates a new **System.Exception** based on the previous **Exception** . *Return Value:* An **Exception** with the same error identification code as the previous **Exception** but with a new message. The message to be displayed to the client when the exception is thrown.

GetHtmlErrorMessage

[C#] public string GetHtmlErrorMessage();
[C++] public: String* GetHtmlErrorMessage();

[VB] Public Function GetHtmlErrorMessage() As String
[JScript] public function GetHtmlErrorMessage() : String;

Description

Returns the HTTP error message to send back to the client.
Return Value: The HTTP error message.

GetHttpCode

[C#] public int GetHttpCode();
[C++] public: int GetHttpCode();
[VB] Public Function GetHttpCode() As Integer
[JScript] public function GetHttpCode() : int;

Description

Returns the HTTP error code to send back to the client. If there is a nonzero HTTP code, it is returned. Otherwise, the **System.Exception.InnerException** code is returned. If neither an **InnerException** code nor a nonzero HTTP code is available, the HTTP error code 500 is returned.
Return Value: The HTTP code representing the exception.

HttpFileCollection class (System.Web)

ToString

Description

Provides access to and organizes files uploaded by a client.

Clients encode files and transmit them in the content body using multipart MIME format with an HTTP **Content-Type** header of **multipart/form-data** . ASP.NET extracts the encoded file(s) from the content body into individual members of an **System.Web.HttpFileCollection** . Methods and properties of the **System.Web.HttpPostedFile** class provide access to the contents and properties of each file.

AllKeys

ToString

[C#] public string[] AllKeys {get;}
[C++] public: __property String* get_AllKeys();
[VB] Public ReadOnly Property AllKeys As String ()
[JScript] public function get AllKeys() : String[];

Description

Gets a string array containing the keys (names) of all members in the file collection.

Count

IsReadOnly

Item

ToString

Description

Gets the object with the specified numerical index from the **System.Web.HttpFileCollection** . The index of the item to get from the file collection.

Item

ToString

[C#] public HttpPostedFile this[string name] {get;}
[C++] public: __property HttpPostedFile* get_Item(String* name);
[VB] Public Default ReadOnly Property Item(ByVal name As String) As HttpPostedFile
[JScript] returnValue = HttpFileCollectionObject.Item(name); Gets an individual **System.Web.HttpPostedFile** object from the file collection. This property is overloaded to allow retrieval of objects by either name or numerical index.

Description

Gets the object with the specified name from the file collection. Name of item to be returned.

Keys

CopyTo

[C#] public void CopyTo(Array dest, int index);
[C++] public: void CopyTo(Array* dest, int index);
[VB] Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)
[JScript] public function CopyTo(dest : Array, index : int);

Description

Copies members of the file collection to an **System.Array** beginning at the specified index of the array. The destination **System.Array**. The index of the destination array where copying starts.

Get

[C#] public HttpPostedFile Get(int index);
[C++] public: HttpPostedFile* Get(int index);
[VB] Public Function Get(ByVal index As Integer) As HttpPostedFile
[JScript] public function Get(index : int) : HttpPostedFile;

Description

Returns the **System.Web.HttpPostedFile** object with the specified numerical index from the file collection. The index of the object to be returned from the file collection.

Get

[C#] public HttpPostedFile Get(string name);
[C++] public: HttpPostedFile* Get(String* name);
[VB] Public Function Get(ByVal name As String) As HttpPostedFile
[JScript] public function Get(name : String) : HttpPostedFile; Returns an individual **System.Web.HttpPostedFile** object from a file collection. This property is overloaded to allow retrieval of objects by either name or numerical index.

Description

Returns the **System.Web.HttpPostedFile** object with the specified name from the file collection. The name of the object to be returned from a file collection.

GetKey

[C#] public string GetKey(int index);
[C++] public: String* GetKey(int index);
[VB] Public Function GetKey(ByVal index As Integer) As String
[JScript] public function GetKey(index : int) : String;

Description

Returns the name of the **System.Web.HttpFileCollection** member with the specified numerical index. The index of the object name to be returned.

HttpModuleCollection class (System.Web)

ToString

Description

Provides a means of indexing and retrieving a collection of **System.Web.IHttpModule** objects.

AllKeys

ToString

```
[C#] public string[] AllKeys {get;}
[C++] public: __property String* get_AllKeys();
[VB] Public ReadOnly Property AllKeys As String ()
[JScript] public function get AllKeys() : String[];
```

Description

Gets a string array containing all the keys (module names) in the **System.Web.HttpModuleCollection** .

Count

IsReadOnly

Item

ToString

Description

Gets the **System.Web.IHttpModule** object with the specified numerical index from the **System.Web.HttpModuleCollection** . The index of the **System.Web.IHttpModule** object to retrieve from the collection.

Item

ToString

```
[C#] public IHttpModule this[string name] {get;}
[C++] public: __property IHttpModule* get_Item(String* name);
[VB] Public Default ReadOnly Property Item(ByVal name As String) As
```

IHttpModule

[JScript] returnValue = HttpModuleCollectionObject.Item(name); Gets the **System.Web.IHttpModule** object with the specified name from the **System.Web.HttpModuleCollection** . This property is overloaded to allow retrieval of modules by either name or numerical index.

Description

Gets the **System.Web.IHttpModule** object with the specified name from the **System.Web.HttpModuleCollection** . Key of the item to be retrieved.

Keys

CopyTo

[C#] public void CopyTo(Array dest, int index);
[C++] public: void CopyTo(Array* dest, int index);
[VB] Public Sub CopyTo(ByVal dest As Array, ByVal index As Integer)
[JScript] public function CopyTo(dest : Array, index : int);

Description

Copies members of the module collection to an **System.Array** beginning at the specified index of the array. The destination **Array**. The index of the destination **Array** where copying starts.

Get

[C#] public IHttpModule Get(int index);
[C++] public: IHttpModule* Get(int index);

[VB] Public Function Get(ByVal index As Integer) As IHttpModule
[JScript] public function Get(index : int) : IHttpModule;

Description

Returns the **System.Web.IHttpModule** object with the specified index from the **System.Web.HttpModuleCollection** . Index of the **System.Web.IHttpModule** object to return from the collection.

Get

[C#] public IHttpModule Get(string name);
[C++] public: IHttpModule* Get(String* name);
[VB] Public Function Get(ByVal name As String) As IHttpModule
[JScript] public function Get(name : String) : IHttpModule; Returns an individual **System.Web.IHttpModule** object from the **System.Web.HttpModuleCollection** . This property is overloaded to allow retrieval of modules by either name or numerical index.

Description

Returns the **System.Web.IHttpModule** object with the specified name from the **System.Web.HttpModuleCollection** . Key of the item to be retrieved.

GetKey

[C#] public string GetKey(int index);
[C++] public: String* GetKey(int index);
[VB] Public Function GetKey(ByVal index As Integer) As String

[JScript] public function GetKey(index : int) : String;

Description

Returns the key (name) of the **System.Web.IHttpModule** object at the specified numerical index.. Index of the key to retrieve from the collection.

HttpParseException class (System.Web)

ToString

Description

The exception that is thrown when a parse error occurs.

HttpParseException

Example Syntax:

ToString

[C#] public HttpParseException(string message, Exception innerException, string fileName, int line);
[C++] public: HttpParseException(String* message, Exception* innerException, String* fileName, int line);
[VB] Public Sub New(ByVal message As String, ByVal innerException As Exception, ByVal fileName As String, ByVal line As Integer)
[JScript] public function HttpParseException(message : String, innerException : Exception, fileName : String, line : int);

Description

Initializes a new instance of the **System.Web.HttpParseException** class. The message displayed to the client when the exception is thrown. The **System.Exception**, if any, that threw the current exception. The name of the file being parsed when the error occurs. The number of the line being parsed when the error occurs.

ErrorCode

FileName

ToString

Description

Gets the name of the file being parsed when the error occurs.

HelpLink

HResult

InnerException

Line

ToString

Description

Gets the number of the line being parsed when the error occurs.

Message

Source

StackTrace

TargetSite

HttpPostedFile class (System.Web)

ToString

Description

Provides a way to access individual files that have been uploaded by a client.

The **System.Web.HttpFileCollection** class provides access to all the files uploaded from a client as a file collection.

ContentLength

ToString

```
[C#]        public      int         ContentLength       {get;}
[C++]       public:     __property  int     get_ContentLength();
[VB]    Public  ReadOnly  Property  ContentLength  As  Integer
[JScript]   public   function   get   ContentLength()   :   int;
```

Description

Gets the size in bytes of an uploaded file.

ContentType

ToString

```
[C#]        public      string      ContentType         {get;}
[C++]       public:     __property  String*     get_ContentType();
[VB]    Public  ReadOnly  Property  ContentType  As  String
```

[JScript] public function get ContentType() : String;

Description

Gets the MIME content type of a file sent by a client.

FileName

ToString

[C#] public string FileName {get;}

[C++] public: __property String* get_FileName();

[VB] Public ReadOnly Property FileName As String

[JScript] public function get FileName() : String;

Description

Gets the fully-qualified name of the file on the client's machine (for example "C:\MyFiles\Test.txt").

InputStream

ToString

[C#] public Stream InputStream {get;}

[C++] public: __property Stream* get_InputStream();

[VB] Public ReadOnly Property InputStream As Stream

[JScript] public function get InputStream() : Stream;

Description

Gets a **System.IO.Stream** object which points to an uploaded file to prepare for reading the contents of the file.

SaveAs

```
[C#]       public     void     SaveAs(string     filename);
[C++]      public:    void     SaveAs(String*    filename);
[VB]   Public   Sub   SaveAs(ByVal   filename   As   String)
[JScript]   public   function   SaveAs(filename   :   String);
```

Description

Saves an uploaded MIME message body to a file on the server. The name of the file.

HttpRequest class (System.Web)

ToString

Description

Enables ASP.NET to read the HTTP values sent by a client during a Web request.

HttpRequest

Example Syntax:

ToString

```
[C#] public HttpRequest(string filename, string url, string queryString);
[C++] public: HttpRequest(String* filename, String* url, String* queryString);
```

[VB] Public Sub New(ByVal filename As String, ByVal url As String, ByVal queryString As String)
[JScript] public function HttpRequest(filename : String, url : String, queryString : String);

Description

Initializes an **System.Web.HttpRequest** object. The name of the file associated with the request. Information regarding the URL of the current request. The entire query string sent with the request (everything after the'?').

AcceptTypes

ToString

[C#] public string[] AcceptTypes {get;}
[C++] public: __property String* get_AcceptTypes();
[VB] Public ReadOnly Property AcceptTypes As String ()
[JScript] public function get AcceptTypes() : String[];

Description

Gets a string array of client-supported MIME accept types.

ApplicationPath

ToString

[C#] public string ApplicationPath {get;}
[C++] public: __property String* get_ApplicationPath();
[VB] Public ReadOnly Property ApplicationPath As String

[JScript] public function get ApplicationPath() : String;

Description

Gets the ASP.NET application's virtual application root path on the server.

Browser

ToString

[C#] public HttpBrowserCapabilities Browser {get; set;}
[C++] public: __property HttpBrowserCapabilities* get_Browser();public: __property void set_Browser(HttpBrowserCapabilities*);
[VB] Public Property Browser As HttpBrowserCapabilities
[JScript] public function get Browser() : HttpBrowserCapabilities;public function set Browser(HttpBrowserCapabilities);

Description

Gets information about the requesting client's browser capabilities.

ClientCertificate

ToString

[C#] public HttpClientCertificate ClientCertificate {get;}
[C++] public: __property HttpClientCertificate* get_ClientCertificate();
[VB] Public ReadOnly Property ClientCertificate As HttpClientCertificate
[JScript] public function get ClientCertificate() : HttpClientCertificate;

Description

Gets the current request's client security certificate.

ContentEncoding

ToString

[C#] public Encoding ContentEncoding {get;}

[C++] public: __property Encoding* get_ContentEncoding();

[VB] Public ReadOnly Property ContentEncoding As Encoding

[JScript] public function get ContentEncoding() : Encoding;

Description

Gets the character set of the entity-body.

Default **ContentEncoding** can be specified in an ASP.NET configuration file. If **ContentEncoding** is specified by the client, the default configuration settings are overridden.

ContentLength

ToString

[C#] public int ContentLength {get;}

[C++] public: __property int get_ContentLength();

[VB] Public ReadOnly Property ContentLength As Integer

[JScript] public function get ContentLength() : int;

Description

Specifies the length, in bytes, of content sent by the client.

ContentType

ToString

[C#] public string ContentType {get;}
[C++] public: __property String* get_ContentType();
[VB] Public ReadOnly Property ContentType As String
[JScript] public function get ContentType() : String;

Description

Gets the MIME content type of the incoming request.

Cookies

ToString

[C#] public HttpCookieCollection Cookies {get;}
[C++] public: __property HttpCookieCollection* get_Cookies();
[VB] Public ReadOnly Property Cookies As HttpCookieCollection
[JScript] public function get Cookies() : HttpCookieCollection;

Description

Gets a collection of cookies sent by the client.

ASP.NET includes two intrinsic cookie collections. The collection accessed through **System.Web.HttpRequest.Cookies** contains cookies transmitted by the client to the server in the **Cookie** header. The collection accessed through **System.Web.HttpResponse.Cookies** contains cookies generated on the server and transmitted to the client in the **Set-Cookie** header.

CurrentExecutionFilePath

ToString

[C#] public string CurrentExecutionFilePath {get;}
[C++] public: __property String* get_CurrentExecutionFilePath();
[VB] Public ReadOnly Property CurrentExecutionFilePath As String
[JScript] public function get CurrentExecutionFilePath() : String;

FilePath

ToString

[C#] public string FilePath {get;}
[C++] public: __property String* get_FilePath();
[VB] Public ReadOnly Property FilePath As String
[JScript] public function get FilePath() : String;

Description

Gets the virtual path of the current request.

The **System.Web.HttpRequest.FilePath** does not include the **System.Web.HttpRequest.PathInfo** trailer. For the URL Http://www.microsoft.com/virdir/page.html/tail, the **FilePath** is Http://www.microsoft.com/virdir/page.html .

Files

ToString

[C#] public HttpFileCollection Files {get;}
[C++] public: __property HttpFileCollection* get_Files();

[VB] Public ReadOnly Property Files As HttpFileCollection
[JScript] public function get Files() : HttpFileCollection;

Description

Gets the collection of client-uploaded files (Multipart MIME format).

The file collection is populated only when the HTTP request Content-Type is multipart/form-data .

Filter

ToString

[C#] public Stream Filter {get; set;}
[C++] public: __property Stream* get_Filter();public: __property void set_Filter(Stream*);
[VB] Public Property Filter As Stream
[JScript] public function get Filter() : Stream;public function set Filter(Stream);

Description

Gets or sets the filter to use when reading the current input stream.

Form

ToString

[C#] public NameValueCollection Form {get;}
[C++] public: __property NameValueCollection* get_Form();
[VB] Public ReadOnly Property Form As NameValueCollection
[JScript] public function get Form() : NameValueCollection;

Description

Gets a collection of form variables.

Populated when the HTTP request Content-Type is either application/x-www-form-urlencoded or multipart/form-data .

Headers

ToString

[C#] public NameValueCollection Headers {get;}

[C++] public: __property NameValueCollection* get_Headers();

[VB] Public ReadOnly Property Headers As NameValueCollection

[JScript] public function get Headers() : NameValueCollection;

Description

Gets a collection of HTTP headers.

HttpMethod

ToString

[C#] public string HttpMethod {get;}

[C++] public: __property String* get_HttpMethod();

[VB] Public ReadOnly Property HttpMethod As String

[JScript] public function get HttpMethod() : String;

Description

Gets the HTTP data transfer method (such as **GET** , **POST** , or **HEAD**) used by the client.

InputStream

ToString

[C#] public Stream InputStream {get;}

[C++] public: __property Stream* get_InputStream();

[VB] Public ReadOnly Property InputStream As Stream

[JScript] public function get InputStream() : Stream;

Description

Gets the contents of the incoming HTTP entity body.

IsAuthenticated

ToString

[C#] public bool IsAuthenticated {get;}

[C++] public: __property bool get_IsAuthenticated();

[VB] Public ReadOnly Property IsAuthenticated As Boolean

[JScript] public function get IsAuthenticated() : Boolean;

Description

Gets a value indicating whether the user has been authenticated.

IsSecureConnection

ToString

[C#] public bool IsSecureConnection {get;}
[C++] public: __property bool get_IsSecureConnection();
[VB] Public ReadOnly Property IsSecureConnection As Boolean
[JScript] public function get IsSecureConnection() : Boolean;

Description

Gets a value indicting whether the HTTP connection uses secure sockets (that is, HTTPS).

Item

ToString

[C#] public string this[string key] {get;}
[C++] public: __property String* get_Item(String* key);
[VB] Public Default ReadOnly Property Item(ByVal key As String) As String
[JScript] returnValue = HttpRequestObject.Item(key);

Description

Default HttpRequest indexed property that retrieves a QueryString, Form, Cookies, or ServerVariables collection. This property is read-only. Numerical index to collection members.

Params

ToString

[C#] public NameValueCollection Params {get;}

[C++] public: __property NameValueCollection* get_Params();
[VB] Public ReadOnly Property Params As NameValueCollection
[JScript] public function get Params() : NameValueCollection;

Description

Gets a combined collection of **System.Web.HttpRequest.QueryString** , **System.Web.HttpRequest.Form** , **System.Web.HttpRequest.ServerVariables** , and **System.Web.HttpRequest.Cookies** items.

Path

ToString

[C#] public string Path {get;}
[C++] public: __property String* get_Path();
[VB] Public ReadOnly Property Path As String
[JScript] public function get Path() : String;

Description

Gets the virtual path of the current request.

The **System.Web.HttpRequest.FilePath** does not include the **System.Web.HttpRequest.PathInfo** trailer. For the URL Http://www.microsoft.com/virdir/page.html/tail, the **FilePath** is Http://www.microsoft.com/virdir/page.html.

PathInfo

ToString

[C#] public string PathInfo {get;}
[C++] public: __property String* get_PathInfo();
[VB] Public ReadOnly Property PathInfo As String
[JScript] public function get PathInfo() : String;

Description

Gets additional path information for a resource with a URL extension.

For the URL Http://www.microsoft.com/virdir/page.html/tail, the **PathInfo** value is /tail.

PhysicalApplicationPath

ToString

[C#] public string PhysicalApplicationPath {get;}
[C++] public: __property String* get_PhysicalApplicationPath();
[VB] Public ReadOnly Property PhysicalApplicationPath As String
[JScript] public function get PhysicalApplicationPath() : String;

Description

Gets the physical file system path of the currently executing server application's root directory.

PhysicalPath

ToString

[C#] public string PhysicalPath {get;}

[C++] public: __property String* get_PhysicalPath();
[VB] Public ReadOnly Property PhysicalPath As String
[JScript] public function get PhysicalPath() : String;

Description

Gets the physical file system path corresponding to the requested URL.

QueryString

ToString

[C#] public NameValueCollection QueryString {get;}
[C++] public: __property NameValueCollection* get_QueryString();
[VB] Public ReadOnly Property QueryString As NameValueCollection
[JScript] public function get QueryString() : NameValueCollection;

Description

Gets the collection of HTTP query string variables.

RawUrl

ToString

[C#] public string RawUrl {get;}
[C++] public: __property String* get_RawUrl();
[VB] Public ReadOnly Property RawUrl As String
[JScript] public function get RawUrl() : String;

Description

Gets the raw URL of the current request.

The raw URL is defined as the part of the URL following the domain information. In the URL string http://www.microsoft.com/articles/recent.aspx, the raw URL is /articles/recent.aspx. The raw URL includes the query string, if present.

RequestType

ToString

[C#] public string RequestType {get; set;}
[C++] public: __property String* get_RequestType();public: __property void set_RequestType(String*);
[VB] Public Property RequestType As String
[JScript] public function get RequestType() : String;public function set RequestType(String);

Description

Gets or sets the HTTP data transfer method (**GET** or **POST**) used by the client.

ServerVariables

ToString

[C#] public NameValueCollection ServerVariables {get;}
[C++] public: __property NameValueCollection* get_ServerVariables();
[VB] Public ReadOnly Property ServerVariables As NameValueCollection
[JScript] public function get ServerVariables() : NameValueCollection;

Description

Gets a collection of web server variables.

TotalBytes

ToString

```
[C#]        public      int         TotalBytes        {get;}
[C++]       public:     __property  int               get_TotalBytes();
[VB]  Public  ReadOnly  Property  TotalBytes  As  Integer
[JScript]   public   function   get   TotalBytes()   :   int;
```

Description

Gets the number of bytes in the current input stream.

Url

ToString

```
[C#]        public      Uri         Url        {get;}
[C++]       public:     __property  Uri*       get_Url();
[VB]  Public  ReadOnly  Property  Url  As  Uri
[JScript]   public   function   get   Url()   :   Uri;
```

Description

Gets Information about the URL of the current request.

UrlReferrer

ToString

[C#] public Uri UrlReferrer {get;}
[C++] public: __property Uri* get_UrlReferrer();
[VB] Public ReadOnly Property UrlReferrer As Uri
[JScript] public function get UrlReferrer() : Uri;

Description

Gets information about the URL of the client's previous request that linked to the current URL.

UserAgent

ToString

[C#] public string UserAgent {get;}
[C++] public: __property String* get_UserAgent();
[VB] Public ReadOnly Property UserAgent As String
[JScript] public function get UserAgent() : String;

Description

Gets the raw user agent string of the client browser.

UserHostAddress

ToString

[C#] public string UserHostAddress {get;}
[C++] public: __property String* get_UserHostAddress();
[VB] Public ReadOnly Property UserHostAddress As String

[JScript] public function get UserHostAddress() : String;

Description

Gets the IP host address of the remote client.

UserHostName

ToString

[C#] public string UserHostName {get;}

[C++] public: __property String* get_UserHostName();

[VB] Public ReadOnly Property UserHostName As String

[JScript] public function get UserHostName() : String;

Description

Gets the DNS name of the remote client.

UserLanguages

ToString

[C#] public string[] UserLanguages {get;}

[C++] public: __property String* get_UserLanguages();

[VB] Public ReadOnly Property UserLanguages As String ()

[JScript] public function get UserLanguages() : String[];

Description

Gets a sorted string array of client language preferences.

BinaryRead

[C#] public byte[] BinaryRead(int count);
[C++] public: unsigned char BinaryRead(int count) __gc[];
[VB] Public Function BinaryRead(ByVal count As Integer) As Byte()
[JScript] public function BinaryRead(count : int) : Byte[];

Description

Performs a binary read of a specified number of bytes from the current input stream.

Return Value: A **byte** array.

The **BinaryRead** method is provided for compatibility with previous versions of ASP. Number of bytes to read.

MapImageCoordinates

[C#] public int[] MapImageCoordinates(string imageFieldName);
[C++] public: int MapImageCoordinates(String* imageFieldName) __gc[];
[VB] Public Function MapImageCoordinates(ByVal imageFieldName As String) As Integer()
[JScript] public function MapImageCoordinates(imageFieldName : String) : int[];

Description

Maps an incoming image-field form parameter to appropriate x/y coordinate values.

Return Value: A two-dimensional array of **integers** . A string reference to a form image map.

MapPath

[C#] public string MapPath(string virtualPath);
[C++] public: String* MapPath(String* virtualPath);
[VB] Public Function MapPath(ByVal virtualPath As String) As String
[JScript] public function MapPath(virtualPath : String) : String; Maps the virtual path in the requested URL to a physical path on the server for the current request.

Description

Maps the specified virtual path to a physical path. The virtual path (absolute or relative) for the current request.

MapPath

[C#] public string MapPath(string virtualPath, string baseVirtualDir, bool allowCrossAppMapping);
[C++] public: String* MapPath(String* virtualPath, String* baseVirtualDir, bool allowCrossAppMapping);
[VB] Public Function MapPath(ByVal virtualPath As String, ByVal baseVirtualDir As String, ByVal allowCrossAppMapping As Boolean) As String
[JScript] public function MapPath(virtualPath : String, baseVirtualDir : String, allowCrossAppMapping : Boolean) : String;

Description

Maps the specified virtual path to a physical path. The virtual path (absolute or relative) for the current request. The virtual base directory path used for relative resolution. If **true**, the *virtualPath* may belong to another application.

SaveAs

[C#] public void SaveAs(string filename, bool includeHeaders);
[C++] public: void SaveAs(String* filename, bool includeHeaders);
[VB] Public Sub SaveAs(ByVal filename As String, ByVal includeHeaders As Boolean)
[JScript] public function SaveAs(filename : String, includeHeaders : Boolean);

Description

Saves an HTTP request to disk.

Saving the request context to disk can be useful in debugging. A string reference to a physical drive path. A **Boolean** value specifying whether an HTTP header should be saved to disk.

HttpResponse class (System.Web)

ToString

Description

Encapsulates HTTP response information from an ASP.NET operation .

The methods and properties of the **HttpResponse** class are exposed through ASP.NET's intrinsic **Response** object.

HttpResponse

Example Syntax:

ToString

[C#] public HttpResponse(TextWriter writer);
[C++] public: HttpResponse(TextWriter* writer);
[VB] Public Sub New(ByVal writer As TextWriter)
[JScript] public function HttpResponse(writer : TextWriter);

Description

Initializes a new instance of the **HttpResponse** class. A **TextWriter** object enabling custom HTTP output.

Buffer

ToString

[C#] public bool Buffer {get; set;}
[C++] public: __property bool get_Buffer();public: __property void set_Buffer(bool);
[VB] Public Property Buffer As Boolean
[JScript] public function get Buffer() : Boolean;public function set Buffer(Boolean);

Description

Gets or sets a value indicating whether to buffer output and send it after the entire response is finished processing.

System.Web.HttpResponse.Buffer has been deprecated in favor of **System.Web.HttpResponse.BufferOutput** and is provided only for compatibility with previous versions of ASP. With ASP.NET, use **System.Web.HttpResponse.BufferOutput** .

BufferOutput

ToString

[C#] public bool BufferOutput {get; set;}

[C++] public: __property bool get_BufferOutput();public: __property void set_BufferOutput(bool);

[VB] Public Property BufferOutput As Boolean

[JScript] public function get BufferOutput() : Boolean;public function set BufferOutput(Boolean);

Description

Gets or sets a value indicating whether to buffer output and send it after the entire page is finished processing.

Cache

ToString

[C#] public HttpCachePolicy Cache {get;}

[C++] public: __property HttpCachePolicy* get_Cache();

[VB] Public ReadOnly Property Cache As HttpCachePolicy

[JScript] public function get Cache() : HttpCachePolicy;

Description

Gets the caching policy (expiration time, privacy, vary clauses) of a Web page.

CacheControl

ToString

[C#] public string CacheControl {get; set;}
[C++] public: __property String* get_CacheControl();public: __property void set_CacheControl(String*);
[VB] Public Property CacheControl As String
[JScript] public function get CacheControl() : String;public function set CacheControl(String);

Description

Sets the **Cache-Control** HTTP header to **Public** or **Private** .

The values for **Private** and **Public** are strings and must be enclosed in quotation marks (" ").

Charset

ToString

[C#] public string Charset {get; set;}
[C++] public: __property String* get_Charset();public: __property void set_Charset(String*);
[VB] Public Property Charset As String

[JScript] public function get Charset() : String;public function set Charset(String);

Description

Gets or sets the HTTP character set of the output stream.

Charset can be set to **null** to suppress the Content-Type header.

ContentEncoding

ToString

[C#] public Encoding ContentEncoding {get; set;}

[C++] public: __property Encoding* get_ContentEncoding();public: __property void set_ContentEncoding(Encoding*);

[VB] Public Property ContentEncoding As Encoding

[JScript] public function get ContentEncoding() : Encoding;public function set ContentEncoding(Encoding);

Description

Gets or sets the HTTP character set of the output stream.

ContentType

ToString

[C#] public string ContentType {get; set;}

[C++] public: __property String* get_ContentType();public: __property void set_ContentType(String*);

[VB] Public Property ContentType As String

[JScript] public function get ContentType() : String;public function set

ContentType(String);

Description

Gets or sets the HTTP MIME type of the output stream.

The following example takes action if the content type of the output is not "Text/HTML".

Cookies

ToString

```
[C#] public HttpCookieCollection Cookies {get;}
[C++] public: __property HttpCookieCollection* get_Cookies();
[VB] Public ReadOnly Property Cookies As HttpCookieCollection
[JScript] public function get Cookies() : HttpCookieCollection;
```

Description

Gets the response cookie collection.

ASP.NET includes two intrinsic cookie collections. The collection accessed through Cookies contains cookies transmitted by the client to the server in the **System.Web.HttpRequest.Cookies** header. The collection accessed through **System.Web.HttpResponse.Cookies** contains cookies generated on the server and transmitted to the client in the **Set-Cookie** header.

Expires

ToString

```
[C#] public int Expires {get; set;}
```

[C++] public: __property int get_Expires();public: __property void set_Expires(int);

[VB] Public Property Expires As Integer

[JScript] public function get Expires() : int;public function set Expires(int);

Description

Gets or sets the number of minutes before a page cached on a browser expires. If the user returns to the same page before it expires, the cached version is displayed.

The **Expires** , **System.Web.HttpResponse.ExpiresAbsolute** and **System.Web.HttpResponse.CacheControl** properties have been deprecated in favor of the methods of the **System.Web.HttpCachePolicy** class available through the **System.Web.HttpResponse.Cache** intrinsic object to control the IIS output cache and client caches.

ExpiresAbsolute

ToString

[C#] public DateTime ExpiresAbsolute {get; set;}

[C++] public: __property DateTime get_ExpiresAbsolute();public: __property void set_ExpiresAbsolute(DateTime);

[VB] Public Property ExpiresAbsolute As DateTime

[JScript] public function get ExpiresAbsolute() : DateTime;public function set ExpiresAbsolute(DateTime);

Description

Gets or sets the absolute date and time at which to remove cached information from the cache.

The **ExpiresAbsolute** , **System.Web.HttpResponse.Expires** and **System.Web.HttpResponse.CacheControl** properties have been deprecated in favor of the methods of the **System.Web.HttpCachePolicy** class available through the **System.Web.HttpResponse.Cache** intrinsic object to control the IIS output cache and client caches.

Filter

ToString

[C#] public Stream Filter {get; set;}
[C++] public: __property Stream* get_Filter();public: __property void set_Filter(Stream*);
[VB] Public Property Filter As Stream
[JScript] public function get Filter() : Stream;public function set Filter(Stream);

Description

Gets or sets a wrapping filter object used to modify the HTTP entity body before transmission.

When you create a **Stream** object and set the **Response.Filter** property to the **Stream** object, all HTTP output sent by **Response.Write** passes through the filter.

IsClientConnected

ToString

[C#] public bool IsClientConnected {get;}
[C++] public: __property bool get_IsClientConnected();
[VB] Public ReadOnly Property IsClientConnected As Boolean
[JScript] public function get IsClientConnected() : Boolean;

Description

Gets a value indicating whether the client is still connected to the server.

Output

ToString

[C#] public TextWriter Output {get;}
[C++] public: __property TextWriter* get_Output();
[VB] Public ReadOnly Property Output As TextWriter
[JScript] public function get Output() : TextWriter;

Description

Enables output of text to the outgoing HTTP response stream.

OutputStream

ToString

[C#] public Stream OutputStream {get;}
[C++] public: __property Stream* get_OutputStream();
[VB] Public ReadOnly Property OutputStream As Stream
[JScript] public function get OutputStream() : Stream;

Description

Enables binary output to the outgoing HTTP content body.

Status

ToString

[C#] public string Status {get; set;}
[C++] public: __property String* get_Status();public: __property void set_Status(String*);
[VB] Public Property Status As String
[JScript] public function get Status() : String;public function set Status(String);

Description

Sets the **Status** line that is returned to the client.

System.Web.HttpResponse.Status has been deprecated in favor of **System.Web.HttpResponse.StatusDescription** and is provided only for compatibility with previous versions of ASP. With ASP.NET, use **System.Web.HttpResponse.StatusDescription** instead.

StatusCode

ToString

[C#] public int StatusCode {get; set;}
[C++] public: __property int get_StatusCode();public: __property void set_StatusCode(int);
[VB] Public Property StatusCode As Integer

[JScript] public function get StatusCode() : int;public function set StatusCode(int);

Description

Gets or sets the HTTP status code of the output returned to the client.

StatusDescription

ToString

[C#] public string StatusDescription {get; set;}
[C++] public: __property String* get_StatusDescription();public: __property void set_StatusDescription(String*);
[VB] Public Property StatusDescription As String
[JScript] public function get StatusDescription() : String;public function set StatusDescription(String);

Description

Gets or sets the HTTP status string of the output returned to the client.

SuppressContent

ToString

[C#] public bool SuppressContent {get; set;}
[C++] public: __property bool get_SuppressContent();public: __property void set_SuppressContent(bool);
[VB] Public Property SuppressContent As Boolean
[JScript] public function get SuppressContent() : Boolean;public function set SuppressContent(Boolean);

Description

Gets or sets a value indicating whether to send HTTP content to the client.

AddCacheItemDependencies

[C#] public void AddCacheItemDependencies(ArrayList cacheKeys);
[C++] public: void AddCacheItemDependencies(ArrayList* cacheKeys);
[VB] Public Sub AddCacheItemDependencies(ByVal cacheKeys As ArrayList)
[JScript] public function AddCacheItemDependencies(cacheKeys : ArrayList);

AddCacheItemDependency

[C#] public void AddCacheItemDependency(string cacheKey);
[C++] public: void AddCacheItemDependency(String* cacheKey);
[VB] Public Sub AddCacheItemDependency(ByVal cacheKey As String)
[JScript] public function AddCacheItemDependency(cacheKey : String);

AddFileDependencies

[C#] public void AddFileDependencies(ArrayList filenames);
[C++] public: void AddFileDependencies(ArrayList* filenames);
[VB] Public Sub AddFileDependencies(ByVal filenames As ArrayList)
[JScript] public function AddFileDependencies(filenames : ArrayList);

Description

Adds a group of file names to the collection of file names on which the current response is dependent. The collection of files to add.

AddFileDependency

[C#] public void AddFileDependency(string filename);
[C++] public: void AddFileDependency(String* filename);
[VB] Public Sub AddFileDependency(ByVal filename As String)
[JScript] public function AddFileDependency(filename : String);

Description

Adds a single file name to the collection of file names on which the current response is dependent. The name of the file to add.

AddHeader

[C#] public void AddHeader(string name, string value);
[C++] public: void AddHeader(String* name, String* value);
[VB] Public Sub AddHeader(ByVal name As String, ByVal value As String)
[JScript] public function AddHeader(name : String, value : String);

Description

Adds an HTTP header to the output stream.

AddHeader is the same as **System.Web.HttpResponse.AppendHeader(System.Web.HttpResponseHeader)** and is provided only for compatibility with previous versions of ASP. With ASP.NET, use **AppendHeader** . The name of the HTTP header to add *value* to. The string to add to the header.

AppendCookie

[C#] public void AppendCookie(HttpCookie cookie);
[C++] public: void AppendCookie(HttpCookie* cookie);
[VB] Public Sub AppendCookie(ByVal cookie As HttpCookie)
[JScript] public function AppendCookie(cookie : HttpCookie);

Description

Adds an HTTP cookie to the intrinsic cookie collection. The cookie to add to the output stream.

AppendHeader

[C#] public void AppendHeader(string name, string value);
[C++] public: void AppendHeader(String* name, String* value);
[VB] Public Sub AppendHeader(ByVal name As String, ByVal value As String)
[JScript] public function AppendHeader(name : String, value : String);

Description

Adds an HTTP header to the output stream.

If you use the **System.Web.HttpResponse.AppendHeader(System.Web.HttpResponseHeader)** method to send cache-specific headers and at the same time use the cache object model (**System.Web.HttpResponse.Cache**) to set cache policy, HTTP response headers pertaining to caching (**Cache-Control** , **Expires** , **Last-Modified** , **Pragma** , and **Vary)** might be deleted when the cache object model is used. This behavior enables ASP.NET to maintain the most restrictive settings. For example, consider a page that includes user controls. If those controls have conflicting cache policies, the most restrictive cache policy will be used. If one user control sets the header " **Cache-Control: Public** " and another sets the more restrictive header " **Cache-Control: Private** " via calls to **System.Web.HttpCachePolicy.SetCacheability(System.Web.HttpCacheability**) , then the " **Cache-Control: Private** " header will be sent with the response. The name of the HTTP header to add to the output stream. The string to append to the header.

AppendToLog

[C#] public void AppendToLog(string param);
[C++] public: void AppendToLog(String* param);
[VB] Public Sub AppendToLog(ByVal param As String)
[JScript] public function AppendToLog(param : String);

Description

Adds custom log information to the IIS log file. The text to add to the log file.

ApplyAppPathModifier

[C#] public string ApplyAppPathModifier(string virtualPath);
[C++] public: String* ApplyAppPathModifier(String* virtualPath);
[VB] Public Function ApplyAppPathModifier(ByVal virtualPath As String) As String
[JScript] public function ApplyAppPathModifier(virtualPath : String) : String;

Description

BinaryWrite

[C#] public void BinaryWrite(byte[] buffer);
[C++] public: void BinaryWrite(unsigned char buffer __gc[]);
[VB] Public Sub BinaryWrite(ByVal buffer() As Byte)
[JScript] public function BinaryWrite(buffer : Byte[]);

Description

Writes a string of binary characters to the HTTP output stream. The bytes to write to the output stream.

Clear

[C#] public void Clear();
[C++] public: void Clear();
[VB] Public Sub Clear()
[JScript] public function Clear();

Description

Clears all content output from the buffer stream.

ClearContent

[C#] public void ClearContent();

```
[C++]        public:      void        ClearContent();
[VB]         Public       Sub          ClearContent()
[JScript]    public       function    ClearContent();
```

Description

Clears all content output from the buffer stream.

ClearHeaders

```
[C#]         public       void        ClearHeaders();
[C++]        public:      void        ClearHeaders();
[VB]         Public       Sub          ClearHeaders()
[JScript]    public       function    ClearHeaders();
```

Description

Clears all headers from the buffer stream.

Close

```
[C#]         public       void        Close();
[C++]        public:      void        Close();
[VB]         Public       Sub          Close()
[JScript]    public       function    Close();
```

Description

Closes the socket connection to a client.

End

[C#] public void End();
[C++] public: void End();
[VB] Public Sub End()
[JScript] public function End();

Description

Sends all currently buffered output to the client, stops execution of the page, and fires the **Application_EndRequest** event.

Flush

[C#] public void Flush();
[C++] public: void Flush();
[VB] Public Sub Flush()
[JScript] public function Flush();

Description

Sends all currently buffered output to the client.

Forces all currently buffered output to be sent to the client.

Pics

[C#] public void Pics(string value);
[C++] public: void Pics(String* value);
[VB] Public Sub Pics(ByVal value As String)
[JScript] public function Pics(value : String);

Description

Appends a **PICS-Label** HTTP header to the output stream.

Platform for Internet Content Selection (PICS) is a World Wide Web Consortium (W3C) standard for content labeling. PICS is essentially a language for creating a ratings system. The string to add to the **PICS-Label** header.

Redirect

[C#] public void Redirect(string url);
[C++] public: void Redirect(String* url);
[VB] Public Sub Redirect(ByVal url As String)
[JScript] public function Redirect(url : String);

Description

Redirects a client to a new URL. The target location.

Redirect

[C#] public void Redirect(string url, bool endResponse);
[C++] public: void Redirect(String* url, bool endResponse);
[VB] Public Sub Redirect(ByVal url As String, ByVal endResponse As Boolean)
[JScript] public function Redirect(url : String, endResponse : Boolean); Redirects a client to a new URL.

RemoveOutputCacheItem

[C#] public static void RemoveOutputCacheItem(string path);

[C++] public: static void RemoveOutputCacheItem(String* path);
[VB] Public Shared Sub RemoveOutputCacheItem(ByVal path As String)
[JScript] public static function RemoveOutputCacheItem(path : String);

SetCookie

[C#] public void SetCookie(HttpCookie cookie);
[C++] public: void SetCookie(HttpCookie* cookie);
[VB] Public Sub SetCookie(ByVal cookie As HttpCookie)
[JScript] public function SetCookie(cookie : HttpCookie);

Description

Updates an existing cookie in the cookie collection.

Write

[C#] public void Write(char ch);
[C++] public: void Write(__wchar_t ch);
[VB] Public Sub Write(ByVal ch As Char)
[JScript] public function Write(ch : Char);

Description

Writes a character to an HTTP output content stream. The character to write to the HTTP output stream.

Write

[C#] public void Write(object obj);

[C++] public: void Write(Object* obj);
[VB] Public Sub Write(ByVal obj As Object)
[JScript] public function Write(obj : Object);

Description

Writes an **Object** to an HTTP output content stream. The **Object** to write to the HTTP output stream.

Write

[C#] public void Write(string s);
[C++] public: void Write(String* s);
[VB] Public Sub Write(ByVal s As String)
[JScript] public function Write(s : String); Writes information to an HTTP output content stream.

Description

Writes a string to an HTTP output content stream. The string to write to the HTTP output stream.

Write

[C#] public void Write(char[] buffer, int index, int count);
[C++] public: void Write(__wchar_t buffer __gc[], int index, int count);
[VB] Public Sub Write(ByVal buffer() As Char, ByVal index As Integer, ByVal count As Integer)
[JScript] public function Write(buffer : Char[], index : int, count : int);

Description

Writes an array of characters to an HTTP output content stream. The character array to write. The position in the character array where writing starts. The number of characters to write, beginning at *index*.

WriteFile

[C#] public void WriteFile(string filename);
[C++] public: void WriteFile(String* filename);
[VB] Public Sub WriteFile(ByVal filename As String)
[JScript] public function WriteFile(filename : String); Writes the specified file directly to an HTTP content output stream.

Description

Writes the specified file directly to an HTTP content output stream. The name of the file to write to the HTTP output.

WriteFile

[C#] public void WriteFile(string filename, bool readIntoMemory);
[C++] public: void WriteFile(String* filename, bool readIntoMemory);
[VB] Public Sub WriteFile(ByVal filename As String, ByVal readIntoMemory As Boolean)
[JScript] public function WriteFile(filename : String, readIntoMemory : Boolean);

Description

Writes the contents of the specified file into a memory block. The name of the file to write into a memory block. Indicates whether the file will be written into a memory block.

WriteFile

[C#] public void WriteFile(IntPtr fileHandle, long offset, long size);
[C++] public: void WriteFile(IntPtr fileHandle, __int64 offset, __int64 size);
[VB] Public Sub WriteFile(ByVal fileHandle As IntPtr, ByVal offset As Long, ByVal size As Long)
[JScript] public function WriteFile(fileHandle : IntPtr, offset : long, size : long);

Description

Writes the specified file directly to an HTTP content output stream. The file handle of the file to write to the HTTP output stream. The byte position in the file where writing will start. The number of bytes to write to the output stream.

WriteFile

[C#] public void WriteFile(string filename, long offset, long size);
[C++] public: void WriteFile(String* filename, __int64 offset, __int64 size);
[VB] Public Sub WriteFile(ByVal filename As String, ByVal offset As Long, ByVal size As Long)
[JScript] public function WriteFile(filename : String, offset : long, size : long);

Description

Writes the specified file directly to an HTTP content output stream. The name of the file to write to the HTTP output stream. The byte position in the file where writing will start. The number of bytes to write to the output stream.

HttpRuntime class (System.Web)

WriteFile

Description

Provides a set of ASP.NET runtime services.

HttpRuntime

Example Syntax:

WriteFile

[C#] public HttpRuntime();
[C++] public: HttpRuntime();
[VB] Public Sub New()
[JScript] public function HttpRuntime();

AppDomainAppId

WriteFile

[C#] public static string AppDomainAppId {get;}
[C++] public: __property static String* get_AppDomainAppId();
[VB] Public Shared ReadOnly Property AppDomainAppId As String
[JScript] public static function get AppDomainAppId() : String;

Description

AppDomainAppPath
WriteFile

[C#] public static string AppDomainAppPath {get;}
[C++] public: __property static String* get_AppDomainAppPath();
[VB] Public Shared ReadOnly Property AppDomainAppPath As String
[JScript] public static function get AppDomainAppPath() : String;

Description

AppDomainAppVirtualPath
WriteFile

[C#] public static string AppDomainAppVirtualPath {get;}
[C++] public: __property static String* get_AppDomainAppVirtualPath();
[VB] Public Shared ReadOnly Property AppDomainAppVirtualPath As String
[JScript] public static function get AppDomainAppVirtualPath() : String;

Description

AppDomainId
WriteFile

```
[C#] public static string AppDomainId {get;}
[C++] public: __property static String* get_AppDomainId();
[VB] Public Shared ReadOnly Property AppDomainId As String
[JScript] public static function get AppDomainId() : String;
```

Description

AspInstallDirectory
WriteFile

```
[C#] public static string AspInstallDirectory {get;}
[C++] public: __property static String* get_AspInstallDirectory();
[VB] Public Shared ReadOnly Property AspInstallDirectory As String
[JScript] public static function get AspInstallDirectory() : String;
```

Description

BinDirectory
WriteFile

```
[C#] public static string BinDirectory {get;}
[C++] public: __property static String* get_BinDirectory();
[VB] Public Shared ReadOnly Property BinDirectory As String
[JScript] public static function get BinDirectory() : String;
```

Description

Cache

WriteFile

```
[C#] public static Cache Cache {get;}
[C++] public: __property static Cache* get_Cache();
[VB] Public Shared ReadOnly Property Cache As Cache
[JScript] public static function get Cache() : Cache;
```

Description

Provides access to the cache.

ClrInstallDirectory

WriteFile

```
[C#] public static string ClrInstallDirectory {get;}
[C++] public: __property static String* get_ClrInstallDirectory();
[VB] Public Shared ReadOnly Property ClrInstallDirectory As String
[JScript] public static function get ClrInstallDirectory() : String;
```

Description

CodegenDir

WriteFile

[C#] public static string CodegenDir {get;}
[C++] public: __property static String* get_CodegenDir();
[VB] Public Shared ReadOnly Property CodegenDir As String
[JScript] public static function get CodegenDir() : String;

Description

IsOnUNCShare
WriteFile

[C#] public static bool IsOnUNCShare {get;}
[C++] public: __property static bool get_IsOnUNCShare();
[VB] Public Shared ReadOnly Property IsOnUNCShare As Boolean
[JScript] public static function get IsOnUNCShare() : Boolean;

Description

MachineConfigurationDirectory
WriteFile

[C#] public static string MachineConfigurationDirectory {get;}
[C++] public: __property static String* get_MachineConfigurationDirectory();
[VB] Public Shared ReadOnly Property MachineConfigurationDirectory As String
[JScript] public static function get MachineConfigurationDirectory() : String;

Description

Close

```
[C#]        public      static      void        Close();
[C++]       public:     static      void        Close();
[VB]        Public      Shared      Sub         Close()
[JScript]   public      static      function    Close();
```

Description

Removes all items from the cache and shuts down the runtime.

ProcessRequest

```
[C#] public static void ProcessRequest(HttpWorkerRequest wr);
[C++] public: static void ProcessRequest(HttpWorkerRequest* wr);
[VB] Public Shared Sub ProcessRequest(ByVal wr As HttpWorkerRequest)
[JScript] public static function ProcessRequest(wr : HttpWorkerRequest);
```

Description

The method that drives all ASP.NET Web processing execution. HttpWorkerRequest object

HttpServerUtility class (System.Web)

ToString

Description

Provides helper methods for processing Web requests.

The methods and properties of the **System.Web.HttpServerUtility** class are exposed through ASP.NET's intrinsic **System.Web.HttpContext.Server** object.

MachineName

ToString

[C#] public string MachineName {get;}
[C++] public: __property String* get_MachineName();
[VB] Public ReadOnly Property MachineName As String
[JScript] public function get MachineName() : String;

Description

Gets the server machine name.

ScriptTimeout

ToString

[C#] public int ScriptTimeout {get; set;}
[C++] public: __property int get_ScriptTimeout();public: __property void set_ScriptTimeout(int);
[VB] Public Property ScriptTimeout As Integer
[JScript] public function get ScriptTimeout() : int;public function set

ScriptTimeout(int);

Description

Gets and sets the request time-out in seconds.

ClearError

[C#] public void ClearError();

[C++] public: void ClearError();

[VB] Public Sub ClearError()

[JScript] public function ClearError();

Description

Clears the previous exception.

CreateObject

[C#] public object CreateObject(string progID);

[C++] public: Object* CreateObject(String* progID);

[VB] Public Function CreateObject(ByVal progID As String) As Object

[JScript] public function CreateObject(progID : String) : Object;

Description

Creates a server instance of a COM object identified by the object's Programmatic Identifier (ProgID).

Return Value: The new object. The class or type of object to be instantiated.

CreateObject

[C#] public object CreateObject(Type type);
[C++] public: Object* CreateObject(Type* type);
[VB] Public Function CreateObject(ByVal type As Type) As Object
[JScript] public function CreateObject(type : Type) : Object;

Description

Instantiates a classic COM object identified via a Type.

CreateObjectFromClsid

[C#] public object CreateObjectFromClsid(string clsid);
[C++] public: Object* CreateObjectFromClsid(String* clsid);
[VB] Public Function CreateObjectFromClsid(ByVal clsid As String) As Object
[JScript] public function CreateObjectFromClsid(clsid : String) : Object;

Description

Creates a server instance of a COM object identified by the object's class identifier (CLSID).

Return Value: The new object. The class identifier of the object to be instantiated.

Execute

[C#] public void Execute(string path);
[C++] public: void Execute(String* path);
[VB] Public Sub Execute(ByVal path As String)
[JScript] public function Execute(path : String); Executes a request to another

page.

Description

Executes a request to another page using the specified URL path to the page.

The **System.Web.HttpServerUtility.Execute(System.String)** method continues execution of the original page after execution of the new page is completed. The **System.Web.HttpServerUtility.Transfer(System.String,System.Boolean)** method unconditionally transfers execution to another page. The URL path of the new request.

Execute

[C#] public void Execute(string path, TextWriter writer);
[C++] public: void Execute(String* path, TextWriter* writer);
[VB] Public Sub Execute(ByVal path As String, ByVal writer As TextWriter)
[JScript] public function Execute(path : String, writer : TextWriter);

Description

Executes a request to another page using the specified URL path to the page. A **System.IO.TextWriter** captures output from the page.

The **System.Web.HttpServerUtility.Execute(System.String)** method continues execution of the original page after execution of the new page is completed. The **System.Web.HttpServerUtility.Transfer(System.String,System.Boolean)** method unconditionally transfers execution to another page. The URL path of the new request. The **System.IO.TextWriter**to capture the output.

GetLastError

[C#] public Exception GetLastError();
[C++] public: Exception* GetLastError();
[VB] Public Function GetLastError() As Exception
[JScript] public function GetLastError() : Exception;

Description

Returns the previous exception.
Return Value: The previous exception that was thrown.

HtmlDecode

[C#] public string HtmlDecode(string s);
[C++] public: String* HtmlDecode(String* s);
[VB] Public Function HtmlDecode(ByVal s As String) As String
[JScript] public function HtmlDecode(s : String) : String; Decodes a string that has been encoded to eliminate illegal HTML characters.

Description

Decodes an HTML-encoded string and returns the decoded string.
Return Value: The decoded text.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The HTML string to decode.

HtmlDecode

[C#] public void HtmlDecode(string s, TextWriter output);
[C++] public: void HtmlDecode(String* s, TextWriter* output);
[VB] Public Sub HtmlDecode(ByVal s As String, ByVal output As TextWriter)
[JScript] public function HtmlDecode(s : String, output : TextWriter);

Description

Decodes an HTML-encoded string and sends the resulting output to a **System.IO.TextWriter** output stream.

URL encoding ensures that all browsers will correctly transmit text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The HTML string to decode. The **System.IO.TextWriter** output stream containing the decoded string.

HtmlEncode

[C#] public string HtmlEncode(string s);
[C++] public: String* HtmlEncode(String* s);
[VB] Public Function HtmlEncode(ByVal s As String) As String
[JScript] public function HtmlEncode(s : String) : String; Encodes a string to be

displayed in a browser.

Description

HTML-encodes a string and returns the encoded string. *Return Value:* The HTML-encoded text.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The text string to encode.

HtmlEncode

[C#] public void HtmlEncode(string s, TextWriter output);
[C++] public: void HtmlEncode(String* s, TextWriter* output);
[VB] Public Sub HtmlEncode(ByVal s As String, ByVal output As TextWriter)
[JScript] public function HtmlEncode(s : String, output : TextWriter);

Description

HTML-encodes a string and sends the resulting output to a **System.IO.TextWriter** output stream.

HTML encoding ensures that text will be correctly displayed in the browser, not interpreted by the browser as HTML. For example, if a text string contains "<" or ">" characters, the browser would interpret these characters as part of HTML tags. The HTML encoding of these two characters is "<" and ">", respectively, which causes the browser to display the angle brackets correctly. The string to encode. The **System.IO.TextWriter** output stream containing the encoded string.

MapPath

```
[C#] public string MapPath(string path);
[C++] public: String* MapPath(String* path);
[VB] Public Function MapPath(ByVal path As String) As String
[JScript] public function MapPath(path : String) : String;
```

Description

Returns the physical file path that corresponds to the specified virtual path on the Web server. *Return Value:* The physical file path that corresponds to *path* . The virtual path on the Web server.

Transfer

```
[C#] public void Transfer(string path);
[C++] public: void Transfer(String* path);
[VB] Public Sub Transfer(ByVal path As String)
[JScript] public function Transfer(path : String);
```

Description

Terminates execution of the current page and begins execution of a new page using the specified URL path to the page. The URL path of the new page on the server to execute.

Transfer

[C#] public void Transfer(string path, bool preserveForm);
[C++] public: void Transfer(String* path, bool preserveForm);
[VB] Public Sub Transfer(ByVal path As String, ByVal preserveForm As Boolean)
[JScript] public function Transfer(path : String, preserveForm : Boolean);
Terminates execution of the current page and begins execution of a new page.

Description

Terminates execution of the current page and begins execution of a new page using the specified URL path to the page. Specifies whether to clear the **System.Web.HttpRequest.QueryString** and **System.Web.HttpRequest.Form** collections. The URL path of the new page on the server to execute. If **true**, the **QueryString** and **Form** collections are preserved. If **false**, they are cleared. The default is **false** .

UrlDecode

[C#] public string UrlDecode(string s);
[C++] public: String* UrlDecode(String* s);
[VB] Public Function UrlDecode(ByVal s As String) As String
[JScript] public function UrlDecode(s : String) : String; Decodes a string encoded for HTTP transmission and sent to the server in a URL.

Description

URL-decodes a string and returns the decoded string. *Return Value:* The decoded text.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The text string to decode.

UrlDecode

[C#] public void UrlDecode(string s, TextWriter output);
[C++] public: void UrlDecode(String* s, TextWriter* output);
[VB] Public Sub UrlDecode(ByVal s As String, ByVal output As TextWriter)
[JScript] public function UrlDecode(s : String, output : TextWriter);

Description

Decodes an HTML string received in a URL and sends the resulting output to a **System.IO.TextWriter** output stream.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The HTML string to decode. The **System.IO.TextWriter** output stream containing the decoded string.

UrlEncode

[C#] public string UrlEncode(string s);
[C++] public: String* UrlEncode(String* s);
[VB] Public Function UrlEncode(ByVal s As String) As String
[JScript] public function UrlEncode(s : String) : String; Encodes a string for reliable HTTP transmission from the Web server to a client via the URL.

Description

URL-encodes a string and returns the encoded string. *Return Value:* The URL encoded text.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The text to URL-encode.

UrlEncode

[C#] public void UrlEncode(string s, TextWriter output);
[C++] public: void UrlEncode(String* s, TextWriter* output);
[VB] Public Sub UrlEncode(ByVal s As String, ByVal output As TextWriter)
[JScript] public function UrlEncode(s : String, output : TextWriter);

Description

URL encodes a string and sends the resulting output to a TextWriter output stream.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The text string to encode. The **System.IO.TextWriter** output stream containing the encoded string.

UrlPathEncode

[C#] public string UrlPathEncode(string s);
[C++] public: String* UrlPathEncode(String* s);
[VB] Public Function UrlPathEncode(ByVal s As String) As String
[JScript] public function UrlPathEncode(s : String) : String; Encodes the path portion of a URL string for reliable HTTP transmission from the Web server to a client via the URL.

Description

URL-encodes the path portion of a URL string and returns the encoded string.

Return Value: The URL encoded text.

URL encoding ensures that all browsers will correctly transmitted text in URL strings. Characters such as "?", "&", "/", and spaces may be truncated or corrupted by some browsers so those characters cannot cannot be used in ASP.NET pages in "" tags or in querystrings where the strings may be sent by a browser in a request string. The text to URL-encode.

HttpStaticObjectsCollection class (System.Web)

UrlPathEncode

Description

Provides a static objects collection for the **System.Web.HttpApplicationState.StaticObjects** property.

HttpStaticObjectsCollection

Example Syntax:

UrlPathEncode

```
[C#] public HttpStaticObjectsCollection();
[C++] public: HttpStaticObjectsCollection();
[VB] Public Sub New()
[JScript] public function HttpStaticObjectsCollection();
```

Count

UrlPathEncode

```
[C#] public int Count {get;}
[C++] public: __property int get_Count();
[VB] Public ReadOnly Property Count As Integer
[JScript] public function get Count() : int;
```

Description

Gets the number of objects in the collection.

IsReadOnly

UrlPathEncode

[C#] public bool IsReadOnly {get;}
[C++] public: __property bool get_IsReadOnly();
[VB] Public ReadOnly Property IsReadOnly As Boolean
[JScript] public function get IsReadOnly() : Boolean;

Description

Gets a value indicating whether the collection is read-only.

IsSynchronized

UrlPathEncode

[C#] public bool IsSynchronized {get;}
[C++] public: __property bool get_IsSynchronized();
[VB] Public ReadOnly Property IsSynchronized As Boolean
[JScript] public function get IsSynchronized() : Boolean;

Description

Gets a value indicating whether the collection is synchronized (i.e.: thread-safe).

Item

UrlPathEncode

[C#] public object this[string name] {get;}
[C++] public: __property Object* get_Item(String* name);

[VB] Public Default ReadOnly Property Item(ByVal name As String) As Object
[JScript] returnValue = HttpStaticObjectsCollectionObject.Item(name);

Description

Gets the object with the specified name from the collection. The case-insensitive name of the object to get.

SyncRoot

UrlPathEncode

[C#] public object SyncRoot {get;}
[C++] public: __property Object* get_SyncRoot();
[VB] Public ReadOnly Property SyncRoot As Object
[JScript] public function get SyncRoot() : Object;

Description

Gets an object that can be used to synchronize access to the collection.

Program code should generally perform synchronized operations on the **SyncRoot** of a collection, not directly on the collection itself. This ensures proper operation of collections that are derived from other objects. Specifically, it maintains proper synchronization with other threads that might be simultaneously modifying the **collection** object.

CopyTo

[C#] public void CopyTo(Array array, int index);
[C++] public: __sealed void CopyTo(Array* array, int index);

[VB] NotOverridable Public Sub CopyTo(ByVal array As Array, ByVal index As Integer)
[JScript] public function CopyTo(array : Array, index : int);

Description

Copies members of an **HttpStaticObjectsCollection** into an array. The array to copy the **HttpStaticObjectsCollection** into. The member of the collection where copying starts.

GetEnumerator

[C#] public IEnumerator GetEnumerator();
[C++] public: __sealed IEnumerator* GetEnumerator();
[VB] NotOverridable Public Function GetEnumerator() As IEnumerator
[JScript] public function GetEnumerator() : IEnumerator;

Description

Returns a dictionary enumerator used for iterating through the key-and-value pairs contained in the collection.
Return Value: The enumerator for the collection.

GetObject

[C#] public object GetObject(string name);
[C++] public: Object* GetObject(String* name);
[VB] Public Function GetObject(ByVal name As String) As Object
[JScript] public function GetObject(name : String) : Object;

Description

Returns the object with the specified name from the collection. This property is an alternative to the **this** accessor. *Return Value:* An object from the collection. The case-insensitive name of the object to return.

HttpUnhandledException class (System.Web)

ToString

Description

The exception that is thrown when a generic exception occurs.

HttpUnhandledException

Example Syntax:

ToString

[C#] public HttpUnhandledException(string message, Exception innerException);
[C++] public: HttpUnhandledException(String* message, Exception* innerException);
[VB] Public Sub New(ByVal message As String, ByVal innerException As Exception)
[JScript] public function HttpUnhandledException(message : String, innerException : Exception); Initializes a new instance of the **System.Web.HttpUnhandledException** class.